Monthly Notices of the ROYAL ASTRONOMICAL SOCIETY

MNRAS **508,** 4017–4037 (2021) https://doi.org/10.1093/mnras/stab2484
Advance Access publication 2021 September 7

ABACUSSUMMIT: a massive set of high-accuracy, high-resolution *N*-body simulations

Nina A. Maksimova,[1] Lehman H. Garrison ,[2]★ Daniel J. Eisenstein,[1] Boryana Hadzhiyska ,[1] Sownak Bose [1] and Thomas P. Satterthwaite[1]

[1]*Center for Astrophysics | Harvard & Smithsonian, 60 Garden Str, Cambridge, MA 02138, USA*
[2]*Center for Computational Astrophysics, Flatiron Institute, 162 5th Ave., New York, NY 10010, USA*

Accepted 2021 August 25. Received 2021 August 19; in original form 2021 April 30

ABSTRACT

We present the public data release of the ABACUSSUMMIT cosmological *N*-body simulation suite, produced with the ABACUS *N*-body code on the Summit supercomputer of the Oak Ridge Leadership Computing Facility. ABACUS achieves $\mathcal{O}(10^{-5})$ median fractional force error at superlative speeds, calculating 70M particle updates per second per node at early times, and 45M particle updates per second per node at late times. The simulation suite totals roughly 60 trillion particles, the core of which is a set of 139 simulations with particle mass $2 \times 10^9\, h^{-1}\, \mathrm{M}_\odot$ in box size $2\, h^{-1}$ Gpc. The suite spans 97 cosmological models, including Planck 2018, previous flagship simulation cosmologies, and a linear derivative and cosmic emulator grid. A subsuite of 1883 boxes of size $500\, h^{-1}$ Mpc is available for covariance estimation. ABACUSSUMMIT data products span 33 epochs from $z = 8$ to 0.1 and include light cones, full particle snapshots, halo catalogues, and particle subsets sampled consistently across redshift. ABACUSSUMMIT is the largest high-accuracy cosmological *N*-body data set produced to date.

Key words: cosmology: theory – methods: numerical.

1 INTRODUCTION

In the next decade, large-scale structure surveys such as DESI (Levi et al. 2013), Euclid (Laureijs et al. 2011), WFIRST (Spergel et al. 2015), LSST (LSST Science Collaboration et al. 2009), and the Subaru Hyper Suprime-Cam (Aihara et al. 2018), and PFS (Tamura et al. 2016) will catalogue tens of millions of galaxies, mapping an unprecedentedly large volume of space. Extracting sub per cent comparisons between survey observations and cosmological predictions, controlling these surveys' systematic errors, and testing their analysis pipelines for cosmological parameter estimation and covariance requires high-precision mock data. Although approximate methods are capable of generating sample data, cosmological *N*-body simulations are a central tool for generating accurate forecasts for the non-linear regime of gravitational structure formation.

Remarkable improvements both in the availability and the performance of computational resources have led to a large increase in the quality and accuracy of cosmological *N*-body simulations. Advances in algorithmic design and high-performance computing have allowed *N*-body codes to simulate more and more particles to better and better mass resolution. Roughly speaking, existing *N*-body suites tend to fall in one of two categories: (1) simulations with a large particle count in one cosmology (up to roughly a few trillion particles) e.g. the Euclid Flagship simulations (Potter & Stadel 2016), OuterRim (Habib et al. 2016; Heitmann et al. 2019), Millenium XXL (Angulo et al. 2012), DarkSky (Skillman et al. 2014), Uchuu (Ishiyama et al. 2020), TianNu (Emberson et al. 2017), Horizon (Kim et al. 2009), or (2) larger suites with a smaller number of particles per box, spanning a broader set of cosmologies, such as MultiDark (Klypin et al. 2016), CoyoteUniverse (Heitmann et al. 2009), Quijote (Villaescusa-Navarro et al. 2020), MICE (Crocce et al. 2010), AbacusCosmos (Garrison et al. 2018a), DarkQuest (Nishimichi et al. 2019), and Aemulus (DeRose et al. 2019).

In anticipation of the forthcoming next generation of large-scale structure surveys, we are releasing a massive suite of high-accuracy *N*-body simulations produced with the ABACUS code on the Summit supercomputer at Oak Ridge National Laboratory. ABACUS is a cosmological *N*-body code that is capable of performing trillions of direct pairwise force calculations per second per node with a median $\mathcal{O}(10^{-5})$ fractional error on force accuracy. ABACUS owes its superlative performance to two key features: (1) a novel algorithm that solves the Poisson equation to great accuracy and (2) performance-optimized, GPU-accelerated software engineering. ABACUS is described in a companion paper to this article (Garrison et al.); previous works on ABACUS include Garrison et al. (2016, 2018a) and Garrison, Eisenstein & Pinto (2018b).

This simulation suite, named ABACUSSUMMIT, consists of 150 simulation boxes, spanning 97 cosmological models, most of which have 330 billion particles with mass $2 \times 10^9 h^{-1}\, \mathrm{M}_\odot$. Another 26 boxes provide re-simulation at worse mass resolution for resolution studies and prototyping, and we provide 1883 smaller boxes at the base resolution for investigation of statistical errors. In total, the suite comprises nearly 60 trillion particles. ABACUSSUMMIT delivers a diversity of publicly available data products designed to cater to a wide variety of science applications, including full particle snapshots, particle subsamples, halo catalogues, particle light cones, projected density maps, and particle IDs for forming high-accuracy halo merger trees.

★ E-mail: lgarrison@flatironinstitute.org

The structure of this paper is as follows. In Section 2, we lay out the suite specifications of AbacusSummit, including the relevant code parameters, data products for each simulation, the chosen cosmologies, and the box realizations. In Section 3, we describe the underlying methods used to produce AbacusSummit: the Abacus *N*-body code, its force solver algorithm, serial and parallel implementations, custom halo-finder CompaSO, on-the-fly light cone implementation, as well as time-stepping, data processing, and force softening methods. In Section 4, we provide a technical review of the performance of Abacus on Summit. In Section 5, we present an overview of cosmological opportunities with the simulations. We summarize in Section 6.

This article is being published as part of a series of papers describing Abacus and AbacusSummit. The Abacus code is described in Garrison et al. (2021a), the CompaSO halo finder in Hadzhiyska et al. (2021), and the merger trees in Bose et al. (2021). With this paper, we are releasing the AbacusSummit simulations for unrestricted public use.

2 AbacusSummit: Suite Specifications

2.1 Overview

The AbacusSummit suite was designed to support a broad range of science applications. This directly informed the choices of cosmological parameters, individual simulation realizations, and data products that comprise the suite. Our main goals were to produce a set of high-accuracy, large-volume simulations of Planck 2018 Lambda cold dark matter (ΛCDM; Aghanim et al. 2018), to generate a broad simulation set in secondary cosmologies for exploring cosmological parameter space, and to provide additional simulations sets designed for comparing *N*-body results across different code implementations, resolutions, and/or parameter choices. A visual summary of a large subset of the simulations is given in Fig. 1.

AbacusSummit begins with a large volume ($400\,h^{-3}\,\mathrm{Gpc}^3$) of high-accuracy simulations of the Planck 2018 ΛCDM cosmological model, separated into 25 base simulations (each of box side $2\,h^{-1}$ Gpc). These simulations each have $6912^3 \approx 330$ billion particles with a particle mass of $2 \times 10^9\,h^{-1}\,\mathrm{M}_\odot$ and force softening of $7.2\,h^{-1}$ proper kpc. We also used this cosmological model for our mass resolution and volume studies. To study group finding and its dependence on mass resolution, we provide a 6 × higher mass resolution simulation in a $1\,h^{-1}$ Gpc box. On the other end of the resolution spectrum, we provide two boxes of $7.5\,h^{-1}$ Gpc size with 27 × lower mass resolution simulation in order to output full-sky light cones to $z > 2$.

To explore cosmological-model dependence, we produced simulations of 96 other cosmologies, nearly all at the base box size and mass resolution. For four secondary cosmologies, we provide six simulations each, phase-matched to the first six of the primary cosmology boxes. We provide eight simulations matched to the cosmological models of other flagship simulations, as well as 5 simulations chosen to explore our treatment of massive neutrinos. Finally, we provide simulations of 79 other cosmologies, all matched to the first of the primary cosmology boxes, to support interpolation in an eight-dimensional parameter space, including ω_0, ω_a, $N_{\rm eff}$, and running of the spectral index.

We provide a suite of 1883 small ($500\,h^{-1}$ Mpc) boxes at the base mass resolution to support studies of statistical errors and covariance matrix estimation under periodic boundary conditions. Finally, AbacusSummit includes six simulations with fixed-amplitude white noise in small boxes ($1185\,h^{-1}$ Mpc, with the base mass resolution), with two of those in a fixed-and-paired doublet (Angulo & Pontzen 2016).

Beyond AbacusSummit proper, we ran several scale-free and high-redshift simulations. The scale-free cosmologies span spectral indices $n_s = -1.5, -2, -2.25$, and -2.5, using $N = 4096^3$ except for the steepest spectral index, which used $N = 6144^3$. The normalization and output choices for these simulations follow Joyce, Garrison & Eisenstein (2021). The high-redshift simulations all employed $N = 6144^3$ in three different box sizes, 20, 80 and $300h^{-1}$ Mpc, terminating at $z = 12$, 8, and 3.5, respectively, making them well-suited for reionization studies. Analysis of these simulations will be presented in future work.

For each simulation, we output extensive data products, including particle subsamples, halo catalogues, merger trees, kernel density estimates, light cones, and projected density maps of the light cone. The details of the available data products are described in Section 2.6.

The AbacusSummit simulation suite was produced using Abacus on the Summit supercomputer at the Oak Ridge Leadership Computing Facility (OLCF). Totaling nearly 60 trillion particles simulated, AbacusSummit is the largest high-accuracy cosmological *N*-body data set produced to date. With this paper, we are placing the data set into the public domain.

Data access is available through OLCF's Constellation portal. The persistent DOI describing the data release is 10.13139/OLCF/1811689. Instructions for accessing the data are given at https://abacussummit.readthedocs.io/en/latest/data-access.html.

In this section, we describe the contents of AbacusSummit, including its full set of cosmologies, individual box realizations, and available data products, as well as notable code choices such as in the generation of initial conditions, time-stepping, and force softening.

2.2 AbacusSummit cosmologies

The cosmologies represented in AbacusSummit are specified as a set of parameters that are then used to create CLASS input files and `abacus.par` parameter files for each simulation run. All cosmologies use $\tau = 0.0544$ and use HyRec to model recombination. The simulations largely span an 8-dimensional cosmological parameter space, parametrized as the CDM density, the baryon density, the Hubble constant, the spectral tilt, the amplitude of structure (i.e. σ_8), the equation of state of dark energy $w(z) = w_0 + (1 - a)w_a$, the running of the spectral tilt, and the density of massless relics $N_{\rm eff}$. All simulations use a flat spatial curvature. In most cases, H_0 is chosen to match the CMB acoustic scale θ_* ($100\theta_* = 1.041533$).

Most of the cosmological models include a single species 60 meV neutrinos, such that $\Omega_\nu = 0.00064420$. A few models use zero or two such species, and one uses a 100 meV species to match an example in the MassiveNuS suite (Liu et al. 2018). We model massive neutrinos as a smooth component, including their effect in the Hubble expansion rate but ignoring the gravitational forces from clustered neutrinos. We set the initial conditions by using CLASS to predict the $z = 1.0$ power spectrum, combining cold dark matter and baryons, and then scale that power spectrum back to the initial redshift of $z = 99$ using the linear growth function including the suppression of growth from the smooth massive neutrino component. In this way, the simulation will arrive at $z = 1$ with the correct linear theory power spectrum for the non-neutrino component. As we limit our suite to low neutrino masses, the free-streaming scale is large, such that treating this component as smooth is a good approximation on

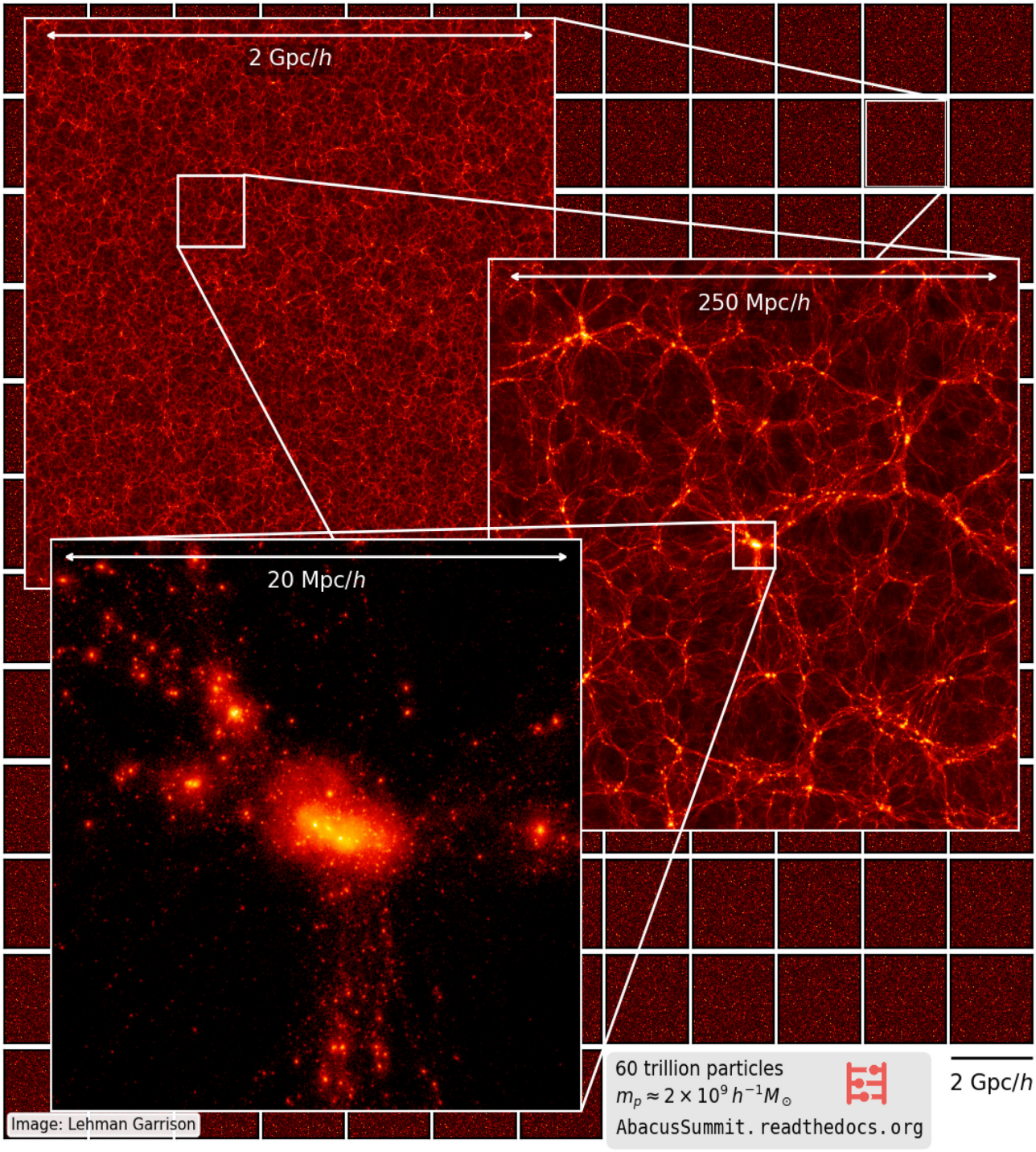

Figure 1. A visualization of the AbacusSummit base-resolution boxes, showing progressive zoom-ins from the full box down to the cluster scale. The 139 boxes that comprise the base-resolution suite are shown as tiles in the background. The renderings display the `AbacusSummit_base_c000_ph000` simulation at $z = 0.1$, and projections are 10 Mpc h^{-1} deep.

small scales and earlier times. Effectively our approximation is to slightly overnormalize the large scales in the initial conditions and then have those scales grow mildly slower than the correct dynamics with clustered neutrinos, arriving at $z = 1$ with the correct linear-regime power. Meanwhile, on small scales where the neutrinos are smoother, except at low redshift around the most massive clusters, our approximation will be more correct.

The full range of AbacusSummit simulated cosmologies is illustrated in Fig. 2 and enumerated in Table 1. Each cosmology has an identifier with three digits, such as `c000` for the Planck 2018 cosmology. This identifier is included in the simulation name. Different realizations of the same cosmology are similarly labelled by a three-digit or four-digit phase number, included as `ph000` in the simulation name.

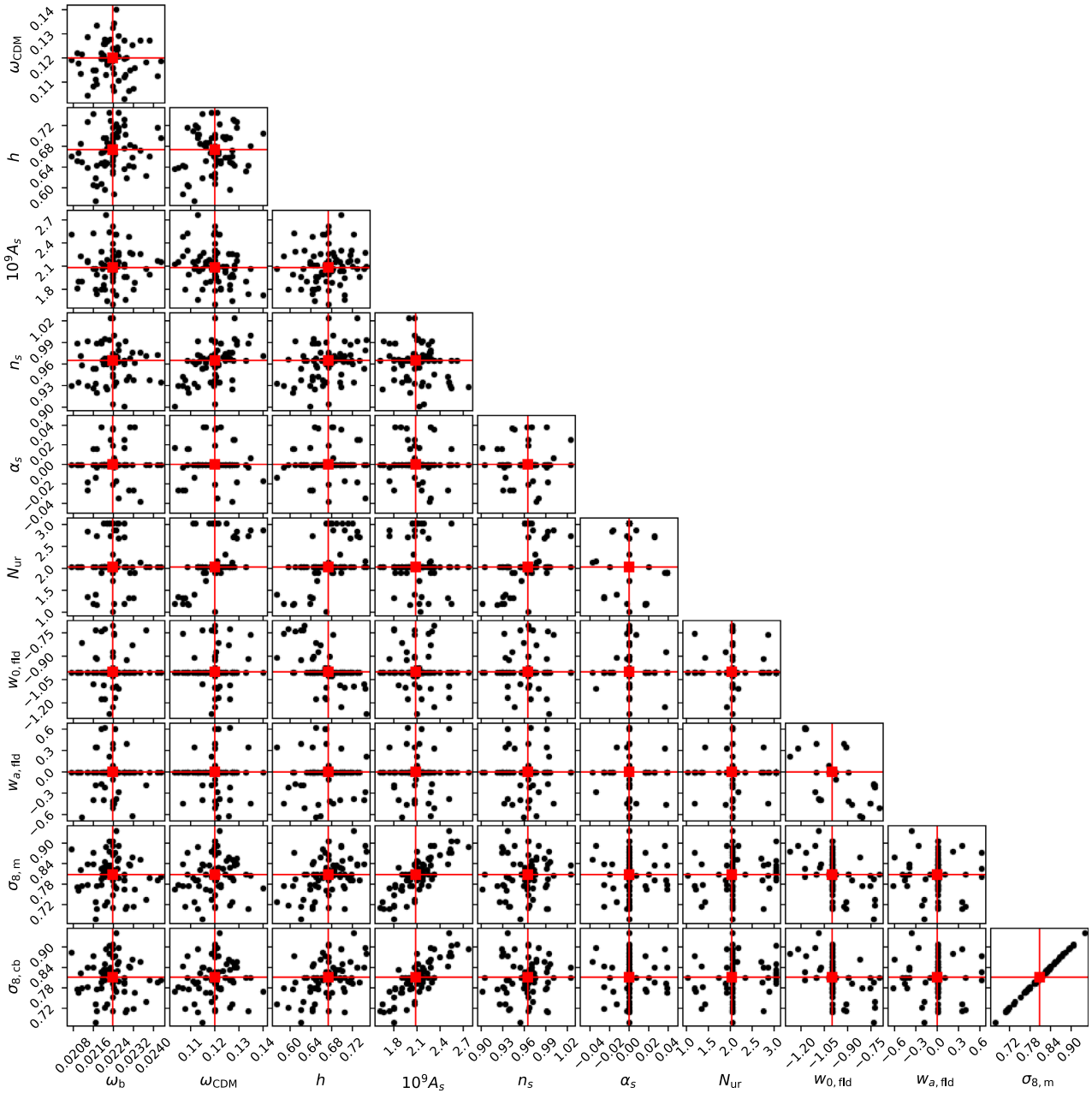

Figure 2. A corner plot representation of the cosmologies spanned by ABACUSSUMMIT. The red square marks the fiducial base cosmology (c000).

Table 1. Cosmological parameters for the first five cosmologies; the remaining 94 are available as an electronic table.

Name	Description	ω_b	ω_c	h	$10^9 A_s$	n_s	α_s	N_{ur}	N_ncdm	$10^4\omega_{ncdm}$	$w_{0,fld}$	$w_{a,fld}$	$\sigma_{8,m}$	$\sigma_{8,cb}$
c000	Baseline ΛCDM	0.02237	0.1200	0.6736	2.0830	0.9649	0.0	2.0328	1	6.4420	−1.0	0.0	0.807952	0.811355
c001	Low ω_c ΛCDM	0.02242	0.1134	0.7030	2.0376	0.9638	0.0	2.0328	1	6.4420	−1.0	0.0	0.776779	0.780222
c002	Thawing dark energy	0.02237	0.1200	0.6278	2.3140	0.9649	0.0	2.0328	1	6.4420	−0.7	−0.5	0.808189	0.811577
c003	$N_{\rm eff} = 3.70$	0.02260	0.1291	0.7160	2.2438	0.9876	0.0	2.6868	1	6.4420	−1.0	0.0	0.855190	0.858583
c004	$\sigma_{8,m} = 0.75$ ΛCDM	0.02237	0.1200	0.6736	1.7949	0.9649	0.0	2.0328	1	6.4420	−1.0	0.0	0.749999	0.753159

The cosmologies are summarized as follows:

c000: Our chosen fiducial cosmology, matching the Planck 2018 results (Aghanim et al. 2018), specifically the mean estimates of the TT, TE, EE+lowE+lensing likelihood chains. In this cosmology, we have run 25 base mass resolution boxes, the 1883 small boxes, and an additional several boxes at both lower and higher resolutions. c009 is the same cosmology, but with massless neutrinos.

c001-c004: Four secondary cosmologies, each with six base boxes, including a low ω_c based on WMAP9 + ACT + SPT (Cal-

abrese et al. 2017), a thawing dark energy model ($w_0 = -0.7$, $w_a = -0.5$), a model with extra relativistic density ($N_{\rm eff} = 3.7$, taken from the `base_nnu_plikHM_TT_lowl_lowE_Riess18_post_BAO` chain of Planck 2018), and a model with lower amplitude clustering (`c000` but with $\sigma_8 = 0.75$).

`c010`: Reference cosmology for our previous AbacusCosmos suite (Garrison et al. 2018a).

`c012-c018`: Reference cosmologies chosen to match existing flagship simulations and allow for future code comparisons, such as the Euclid Flagship runs, OuterRim, DarkSky, Horizon, IllustrisTNG, and MultiDark (Habib et al. 2016; Heitmann et al. 2019; Skillman et al. 2014; Dubois et al. 2016; Nelson et al. 2019; Klypin et al. 2016).

`c019-c020`: Variations around `c000` using zero or two massless neutrinos, varying H_0 so as to hold the CMB acoustic scale θ_* fixed.

`c021-c022`: Matches to two MassiveNUs simulations (Liu et al. 2018), specifically the massless and single 100 meV models.

`c100-c116`: A linear derivative grid with eight matched pair cosmologies, with small positive and negative steps away from `c000` in an eight-dimensional parameter space, as well as one additional model `c116` that is the high σ_8 matched pair to the low σ_8 secondary cosmology `c004`. The cosmological model space is parametrized as the cold dark matter density ($\omega_c = \Omega_c h^2$), the baryon density ($\omega_b = \Omega_b h^2$), the spectral tilt n_s, the amplitude of structure σ_8 of the CDM and baryons, the equation of state of dark energy $w(z) = w_0 + (1 - a)w_a$, the running of the spectral tilt α_s, and the density of massless relics $N_{\rm eff}$. In detail, we vary w_a holding the equation of state at $z = 0.333$ constant, so that a change in w_a is accompanied by a shift in w_0 that is –1/4 as big; this was done to allow larger variations in w_a while keeping the low-redshift cosmic distance scale less changed. Similarly, changes in $N_{\rm eff}$ are compensated by changes in ω_c and n_s, based on the covariances in the Markov Chain posteriors in Planck, so as to leave the CMB less changed. All simulations use a flat spatial curvature, and H_0 is chosen to match the CMB acoustic scale θ_* to that of `c000`.

`c117-c126`: Another linear derivative grid with smaller steps, in ω_c, n_s, w_0, w_a, and σ_8, to provide more exact linear derivatives for small alterations of the base `c000` model.

`c130-c181`: A larger unstructured emulator grid that provides a wider coverage of the eight-dimensional parameter space. In detail, this was done by placing the points on the surface (not interior) of an eight-dimensional ellipsoid, whose extent was chosen to be relatively generous (5–8 standard deviations in any one parameter) beyond today's constraints from the combination of CMB and large-scale structure data. After the point on the ellipsoid is set, an additional uniform random excursion between –0.06 and +0.06 is added to σ_8; this direction was seen as special as other aspects of a cosmological model (such as the optical depth τ) might impact the amplitude of low-redshift clustering while not altering the shape of the power spectrum as much.

The spread of points on the eight-dimensional ellipsoid was constructed to spread the points out, subject to certain constraints, and to avoid antipodal reflections that would by parity not add new information to the computation of second derivatives. Because emulators might not want to use the full eight-dimensional space, we opt to add constraints so that smaller subsets would explore only simpler spaces. There are four sets of points. The first varies only ω_c, ω_b, n_s, and σ_8; the second adds w_0 and w_a; the third adds α_s and $N_{\rm eff}$; and the fourth uses all eight-dimensions. In the first set, we do positive and negative excursions purely in the ω_c, n_s, and σ_8 directions and then have 11 additional points. We constructed a glass on a unit sphere, subject to these constraints, by starting with random points on the unit sphere, with their antipodes included, and then evolving by an electrostatic repulsion. The antipodes were discarded in all but three pairs that were held at the unit vector in the ω_c, n_s, and σ_8 directions. These three pairs were also excluded from the extra uniform spread in σ_8.

2.3 AbacusSummit simulations

For AbacusSummit, we then performed simulations of these cosmological models in a variety of configurations. For each box size, we may perform simulations with multiple realizations of the initial conditions. The random seed of the white noise of the Gaussian random field is labeled by `ph000-ph4999`. Hence, simulations with the same phase seed but with different mass resolution or cosmological parameters will share the same initial conditions white noise; such simulations are always of the same box size. Matching the simulation initial conditions in white noise allows comparison between cosmologies and across mass resolutions with substantially suppressed sample variance. In particular, this facilitates the construction of derivatives of observables with respect to cosmological parameters.

`base`: The bulk of the computational resource is in this configuration, 6912^3 particles in $2h^{-1}$ Gpc box. For `c000`, this corresponds to a particle mass of $2 \times 10^9 h^{-1}\,{\rm M}_\odot$; this mass will vary slightly in other cosmologies. There are 139 of these simulations.

`high`: A single simulation with 6 times better mass resolution, about $3 \times 10^8 h^{-1}\,{\rm M}_\odot$, using 6300^3 particles in a $1h^{-1}$ Gpc box.

`highbase`: Three simulations with the base particle mass but in $1h^{-1}$ Gpc, hence 3456^3 particles.

`huge`: Two simulations with 8640^3 particles in $7.5h^{-1}$ Gpc boxes, implying 27-fold worse particle mass resolution, around $5 \times 10^{10} h^{-1}\,{\rm M}_\odot$.

`hugebase`: Twenty-five simulations of $2h^{-1}$ Gpc boxes with 2304^3 particles, matching the mass resolution of `huge`.

`small`: Simulations of the base mass resolution in $500h^{-1}$ Mpc boxes, with 1728^3 particles. 2000 of these simulations were initiated, but some crashed for reasons unrelated to the density field. We present 1883 of these boxes, 1643 of which reached the final time.

`fixedbase`: Simulations of 4096^3 particles in $1.18h^{-1}$ Gpc boxes, matching the mass resolution of `base`, with white noise chosen to be of fixed amplitude in Fourier space. That is, all Fourier modes have a random phase but a complex norm of unity, multiplied by the power spectrum. There are 6 such simulations, 2 of which share identical white noise but with opposite signs for the mode amplitudes.

The individual simulation realizations comprising the AbacusSummit suite are enumerated in Tables 2 and 3.

2.4 Initial conditions

We generate initial conditions with the public `zeldovich-PLT`[1] code, which uses the first-order particle linear theory corrections described in Garrison et al. (2016). The second-order Lagrangian perturbation theory correction is computed with direct force evaluation during the first two steps of Abacus. To aid methods that seek to reduce large-scale structure sample variance by utilizing cross-correlation with the initial conditions, we make available reduced-resolution versions of the initial displacements and linear density field. These are provided at 576^3 and 1152^3 resolutions

[1] https://github.com/abacusorg/zeldovich-PLT

Table 2. The AbacusSummit simulations in the base Planck 2018 ΛCDM cosmology (c000). The simulation name contains the label corresponding to the specific phase realization of each cosmology (e.g. ph007). PPD refers to the particles per dimension; PPD^3 gives the total number of particles simulated in the box. The length of the cubic box is specified in the 'Size' column. Only a few of our simulations include the full timeslice; typically only subsamples are saved. The 'Full Outputs' column states the redshifts at which timeslices were saved (in addition to subsamples). 'Full' refers to the full list of output redshifts, defined as z = 3.0, 2.5, 2.0, 1.7, 1.4, 1.1, 0.8, 0.5, 0.4, 0.3, 0.2, 0.1. The 'Partial' list is $z = 2.5, 1.4, 0.8, 0.2$. 'Partial+HiZ' adds $z = 3.0$ and 2.0 to that.

Name	PPD	Size (Mpc h^{-1})	$z_{\rm final}$	Full outputs	Description
AbacusSummit_base_c000_ph000	6912	2000	0.1	Full	Planck 2018 ΛCDM
AbacusSummit_base_c000_ph{001-005}	6912	2000	0.1	Partial+HiZ	Planck 2018 ΛCDM
AbacusSummit_base_c000_ph{006-024}	6912	2000	0.1	none	Planck 2018 ΛCDM
AbacusSummit_high_c000_ph100	6300	1000	0.1	Full	High-res ΛCDM
AbacusSummit_highbase_c000_ph100	3456	1000	0.1	Full	Base-res ΛCDM
AbacusSummit_huge_c000_ph{201-202}	8640	7500	0.1	1.4, 1.1, 0.8, 0.5, 0.2	Low-res ΛCDM, box-centred light cone
AbacusSummit_hugebase_c000_ph{000-024}	2304	2000	0.1	1.4, 1.1, 0.8, 0.5, 0.2	Low-res match to base
AbacusSummit_fixed_c000_ph099	4096	1185	0.1	Full	Base-res ΛCDM, fixed amplitude
AbacusSummit_fixed_c000_ph098	4096	1185	0.1	Full	Base-res ΛCDM, phase inverted from ph099
AbacusSummit_small_c000_ph{3000-4999}	1728	500	0.2	none	Base-res ΛCDM small boxes

Table 3. Like Table 2, but for simulations with cosmologies other than the baseline c000 choice. The simulation name encodes the cosmology choice (e.g. c001): a brief description of the cosmology is given in the rightmost column and for more details, refer to Table 1.

Name	PPD	Size (Mpc h^{-1})	$z_{\rm final}$	Full outputs	Description
AbacusSummit_base_c001_ph000	6912	2000	0.1	Partial+HiZ	WMAP7 model with low Ω_c
AbacusSummit_base_c001_ph{001-005}	6912	2000	0.1	Partial	WMAP7
AbacusSummit_fixed_c001_ph099	4096	1185	0.1	Full	WMAP7, fixed amplitude
AbacusSummit_base_c002_ph000	6912	2000	0.1	Partial+HiZ	Thawing wCDM $w_0 = -0.7$, $w_a = -0.5$
AbacusSummit_base_c002_ph{001-005}	6912	2000	0.1	Partial	Thawing wCDM
AbacusSummit_fixed_c002_ph099	4096	1185	0.1	Full	Thawing wCDM, fixed amplitude
AbacusSummit_base_c003_ph000	6912	2000	0.1	Partial+HiZ	$N_{\rm eff} = 3.70$ model
AbacusSummit_base_c003_ph{001-005}	6912	2000	0.1	Partial	$N_{\rm eff} = 3.70$ model
AbacusSummit_fixed_c003_ph099	4096	1185	0.1	Full	$N_{\rm eff} = 3.70$ model, fixed amplitude
AbacusSummit_base_c004_ph000	6912	2000	0.1	Partial+HiZ	Low $\sigma_{8,\,\rm matter} = 0.75$, otherwise baseline ΛCDM
AbacusSummit_base_c004_ph{001-005}	6912	2000	0.1	Partial	Low $\sigma_{8,\,\rm matter} = 0.75$ ΛCDM
AbacusSummit_fixed_c004_ph099	4096	1185	0.1	Full	$\sigma_{8,\,\rm matter} = 0.75$ ΛCDM, fixed amplitude
AbacusSummit_base_c009_ph000	6912	2000	0.1	Partial	Baseline ΛCDM with massless neutrinos
AbacusSummit_base_c010_ph000	6912	2000	0.1	None	AbacusCosmos LCDM
AbacusSummit_base_c012_ph000	6912	2000	0.1	None	Euclid Flagship1
AbacusSummit_base_c013_ph000	6912	2000	0.1	None	Euclid Flagship2
AbacusSummit_base_c014_ph000	6912	2000	0.1	None	OuterRim
AbacusSummit_base_c015_ph000	6912	2000	0.1	None	Dark Sky
AbacusSummit_base_c016_ph000	6912	2000	0.1	None	Horizon
AbacusSummit_base_c017_ph000	6912	2000	0.1	None	Illustris
AbacusSummit_base_c018_ph000	6912	2000	0.1	None	Multidark Planck
AbacusSummit_base_c019_ph000	6912	2000	0.1	None	Baseline ΛCDM w/two 60 meV neutrino species
AbacusSummit_base_c020_ph000	6912	2000	0.1	None	Baseline ΛCDM w/massless neutrinos
AbacusSummit_highbase_c021_ph000	3456	1000	0.1	Partial	MassiveNUs base model with massless neutrinos
AbacusSummit_highbase_c021_ph000	3456	1000	0.1	Partial	MassiveNUs with 100 meV neutrino species
AbacusSummit_base_c{100-115}_ph000	6912	2000	0.1	None	Linear Derivative Grid
AbacusSummit_base_c116_ph000	6912	2000	0.1	None	Linear Derivative Grid
AbacusSummit_base_c{117-126}_ph000	6912	2000	0.1	None	Finer Linear Derivative Grid
AbacusSummit_base_c{130-181}_ph000	6912	2000	0.1	None	Broader Emulator Grid

and only include the first-order (ZA) part. The lower 10 bits of the displacements and densities are truncated to aid compression, a modification of < 0.01 per cent. The displacements and densities are stored in separate files, the headers of which include the simulation parameters, the full CLASS power spectrum, and table of growth factors as computed by Abacus's cosmology module.

We do not provide the full-resolution initial conditions, but we do make available the abacus.par parameter file as well as the CLASS input power spectrum (Lesgourgues 2011), from which the initial conditions may be readily regenerated using zeldovich-PLT.

As described in Garrison et al. (2016), zeldovich-PLT performs a rescaling to correct for the violation of linear theory that occurs on small scales in discrete particle systems. The target correction redshift is chosen to be $z = 12$, and the particles are displaced from an initial cubic grid using the particle lattice eigenmodes. The particles' initial positions relative to the starting grid are permanently encoded as part of their particle IDs.

With the approximation of neutrinos as a smooth, non-clustering component (Section 2.2), the growth rate, and therefore the linear theory velocities, is modified. Defining $f_c = 1 - \Omega_\nu/\Omega_M$ as the clustering fraction, the growth rate correction can ordinarily be applied in configuration space as a multiplicative prefactor of $(\sqrt{1+24f_c}-1)/4$ on the velocity. However, with the PLT correction of a $\mathbf{k}$-dependent eigenvalue, $\epsilon(\boldsymbol{k})$, the f_c and PLT factors must be combined in Fourier space at the mode level, yielding a prefactor of $(\sqrt{1+24\epsilon(\mathbf{k})f_c}-1)/4$ on each velocity mode.

2.5 Code parameters

AbacusSummit simulations use the spline force softening scheme described in Garrison et al. (2018b), chosen because the force law returns to exactly the $1/r^2$ form at a scale mildly larger than the softening length. This is desirable because the far-field method computes unsoftened forces, and because it limits the modification of the force law to scales where the two-body scattering timescale is short. In Garrison, Joyce & Eisenstein (2021c), we show that using force softening that is fixed in proper, rather than comoving, distance gives more efficient solutions, reaching mildly better convergence to the small-scale correlation function in scale-free simulation tests while requiring substantially fewer time-steps. The physical reasoning is clear: the collapsed interiors of haloes are not participating in the Hubble expansion and so one should maintain a consistent force law, consistent with the ideas of stable clustering. For our base boxes, we select a Plummer-equivalent softening length of $7.2h^{-1}$ proper kpc, capped at 30 per cent of the interparticle spacing at early times ($z > 11$). Simulations with other mass resolutions scale the softening by the interparticle spacing. Garrison et al. (2021c) shows that one should expect few per cent convergence on the two-point correlation function at about 10 per cent of the interparticle spacing, so about $30h^{-1}$ proper kpc. It also demonstrates that this convergence is not improved by yet smaller softening lengths but is instead determined by the mass resolution; we therefore have chosen to avoid the extra computational cost that the smaller time-steps required by finer softening would incur. Clustering well inside the proper softening length is suppressed relative to if the softening were fixed in comoving distance to the $z = 0$ value, but these are precisely the scales for which neither solution is converged due to the mass resolution limit.

As described in further detail in Section 3, Abacus subdivides the simulation space into a cubic grid with CPD cells per dimension. CPD is chosen solely to balance the computational workload in the near-field and far-field force computations and does not affect the accuracy or physical results of the simulation. The cell structure does appear in some aspects of the data product file organization, as will be described later. The AbacusSummit base simulations set CPD to 1701, yielding an average of 67 particles per cell.

Abacus evolves the particle dynamics using second-order leapfrog integration and global time-stepping, where all particles use the same time-step. While this is imposed by the current structure of the code, it also offers some important advantages, most notably that since the time-step is chosen to serve the densest regions, most of the particles have a rather short time-step and more accurate integrations. The variation of individualized time-steps also tends to break the symplectic property of leapfrog (Quinn et al.), although more complicated formulations can solve this (Farr & Bertschinger 2007); we have not explored whether our global time-stepping does better in this regard. We note that cosmological simulations such as AbacusSummit do not have the mass resolution to properly resolve dynamical resonances, regardless of time-stepping (e.g. Weinberg & Katz 2007).

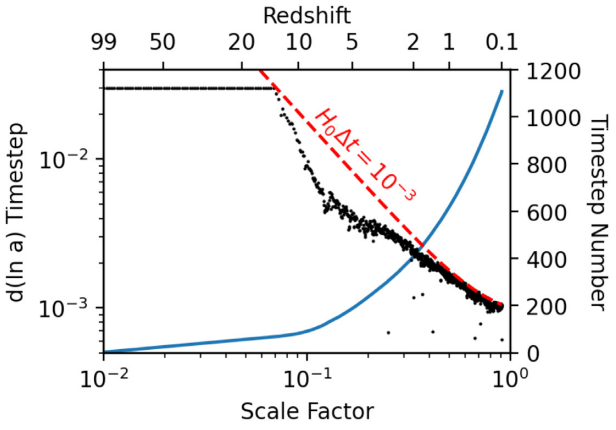

Figure 3. The time-step in the `base_c000_ph000` sim as a function of the scale factor a. Black points show the change in $\ln(a)$, showing that the simulation begins with steps of $\Delta\ln(a) = 0.03$ in the pre-collapse regime, and then drops rapidly at $z = 15$, finishing with steps around 10^{-3}. The few particularly short time steps are due to choosing to arrive at a particular output redshift. The red line shows a line of constant change in proper time; because we use a proper softening length, the simulation tracks this behaviour, with steps of about $0.001H_0^{-1} \approx 10^7 h^{-1}$ yr. The blue line and the scale on the right-hand side shows the cumulative number of time-steps as a function of scale factor.

The time-step selection criteria are described in detail in Garrison et al. (2018b). For most of the steps, we are limited by our small-scale criteria, which is computed as the minimum value over all cells of the ratio of the per-cell velocity dispersion to the per-cell maximum particle acceleration, scaled by a tuning parameter η. For AbacusSummit, we choose η to be 0.25. At early times when the particles are all close to the initial grid, we are limited by a cosmological bound that the global time step $d(\ln a)$, for scale factor a, never exceed 0.03.

The resulting time-step behaviour of a typical base-resolution AbacusSummit simulation (particle mass $2 \times 10^9 h^{-1}\,\mathrm{M}_\odot$) is shown in Fig. 3. We require roughly 1100 steps to reach $z = 0.1$. The time-step in $\ln a$ decreases rapidly at $z = 15$ with the first close approach of pairs of particles (but recall that the simulation is highly softened at early times because of the use of proper softening). At late times, the time-step asymptotes to a value of $0.001H_0^{-1} \approx 10h^{-1}$ Myr. This constant level is expected because of the constant proper softening and the fact that the innermost density of haloes increases only slowly. It is perhaps surprising that this constant-time asymptote is approached from below, not above. We expect this is because the velocity dispersion of the Mpc-scale cell around the dense cores is increasing over time as the bigger haloes build up. The effect is particularly noticeable at redshift 5–10, where the collapsed haloes are often smaller than a cell, so that the velocity dispersion is diluted by the colder circum-halo particles.

2.6 Data products

AbacusSummit consists of a total of roughly 2 PB of available data products. For the purposes of describing output data products, we define 12 primary redshifts and 21 secondary redshifts between $z = 0.1$ and $z = 8.0$. The primary set is $z = 0.1, 0.2, 0.3, 0.4, 0.5, 0.8, 1.1, 1.4, 1.7, 2.0, 2.5$, and 3.0; the secondary set is 0.15, 0.25, 0.35, 0.45, 0.575, 0.65, 0.725, 0.875, 0.95, 1.025, 1.175, 1.25, 1.325, 1.475,

1.55, 1.625, 1.85, 2.25, 2.75, 3.0, 5.0, and 8.0. We have designed a set of data products aimed at supporting mock catalogues to be constructed using halo occupation distributions, as well as efficient access to measurements of the density fields.

For AbacusSummit halo catalogues and related data products, Abacus uses CompaSO: a custom, on-the-fly halo finding algorithm summarized in Section 3.3 and described comprehensively in Hadzhiyska et al. (2021). Using CompaSO, we identify and output halo information at 33 epochs. For the purpose of this section, we note that CompaSO processing results in particles being assigned to disjoint L0 sets, inside of which are found disjoint L1 haloes, inside of which are found disjoint L2 subhaloes. The L1 haloes are the key CompaSO product and for these we output a large set of summary statistics, listed in Table 4, as well as constituent particles.

Beyond haloes, we output a 10 per cent subsample of particles. This is suitable for many applications, e.g. the computation of matter-field statistics such as gravitational lensing observables and the Monte Carlo sampling of the halo mass distribution for the siting of satellite galaxies in halo-occupation modelling. The subsample of particles is consistent between output redshifts and the particle ID numbers are included, so that one can track individual particles in time. At primary redshifts, we output the position, velocity, particle ID, and kernel density estimate of the full subsample of particles.

By connecting the particle ID numbers for a small, consistent subsample of particles in each halo, one can build merger trees by post-processing to find which haloes contain matching sets of particles. We provide merger tree aggregations, described in Bose et al. (2021), that use the IDs to rapidly associate haloes between time slices and build up the progenitor history of the late-time haloes.

The key data products available as part of AbacusSummit are:

(i) CompaSO-identified L1 halo catalogues with comprehensive statistics computed on-the-fly from the full particle set, available at all primary and secondary redshifts.

(ii) Merger tree associations of these haloes, generated from post-processing of the particle IDs between adjacent redshifts. We additionally store halo progenitor information matched across epochs separated by two snapshots, which provides a simple diagnostic for the frequency of transient fly-by events.

(iii) Cleaned halo catalogues, row-matched to the L1 catalogues. The cleaning uses the merger trees to aggregate haloes that are presently separate but that were unified in the past; this appears to redress some examples of overly aggressive deblending of haloes.

(iv) 10 per cent subsamples of particle data, split into 3 per cent and 7 per cent sets, called A and B, so that users can minimize their data access based on application. These sets of particles are consistently defined across redshift and are selected effectively randomly based on a hash of the particle ID number. At primary redshifts, we output the position, velocity, particle ID, and kernel density estimate of the full subsample of particles, split into two sets of files based on whether each particle belongs to an L0 halo or not. For each L0 halo, particles belonging to the same L1 halo are in contiguous sets, and the indices required to access them are stored in the halo catalogue statistics. Particle positions and velocities are output in one file. A separate file contains the unique particle ID, which encodes the initial grid position, as well as the kernel density estimate and a sticky bit that is set if the particle has ever been in the most massive L2 halo of an L1 halo with more than 35 particles. At secondary redshifts, we provide only the particles in L1 haloes and only particle ID file.

(v) A light cone stretching from the corner of the box and including a single second periodic copy of the box. This provides an octant of sky to $z = 0.8$ and about 900 deg^2 to $z = 2.45$. For particles belonging to both the 10 per cent subsample and the light cone at any given epoch, we output their positions, velocities, and IDs, as well as each particle's HEALPix pixel number (in Nested order) used to form projected density maps, where $N_{\rm side} = 16384$.

(vi) In addition to all the above, we also output the full particle set for a few of the primary timeslices of a few boxes, including all positions, velocities, and particle IDs.

(vii) Matter power spectrum measurements using the 10 per cent particle subsample, or the full particle set when available. The $\ell = 0$ real-space and $\ell = 0, 2, 4$ redshift-space spectra are reported.

(viii) The initial density field and corresponding particle displacements at 576^3 and 1152^3 resolutions.

The purpose of the secondary redshifts is to support generation of merger trees, as the particle IDs can be used to associate haloes between snapshots. Therefore, for the 24 secondary redshifts, we output the halo catalogues and the halo subsample particle IDs (which encode each particle's kernel density estimate and its sticky L2 tag) only, but not the positions or velocities of the halo particles, nor any information regarding the field particles.

A particle is tagged if it is taggable and is in the largest L2 halo for a given L1 halo. The particle IDs can likewise be used to associate halo catalogues to the light cones, as the particle subsamples are consistent across epochs. We stress likewise that the particle IDs – which are output for the particle subsample at all secondary redshifts and for the full particle set at the primary redshifts – each contain the kernel density estimate, which in its own right forms a powerful data product since it is output for many particles at a large number of redshifts. It is also possible to use the secondary redshifts to associate halo catalogues to the light cones, as the subsample of particles (and therefore their IDs) are the same.

2.7 File formats and directory structures

We have organized the data products of AbacusSummit into structures that we believe will be highly useful for users while trying to minimize the data volume. Even with these aggressive compressions, the total data volume is nearly 2 PB, and so it is important that most user applications not need to read the entire data set. More detailed documentation is available at https://abacussummit.readthedocs.io; here, we focus on some design aspects that might be useful for other programmes.

Each simulation is located in its own directory, and the bulk of the data is found in four subdirectories: `halo` for the halo catalogues and particle subsamples, `lightcone` for the light cone outputs, `cleaned` for the post-processed cleaned catalogues, and `slices` for the time slices. The next subdirectory level separates the various redshifts. Below that, each type of file is given a separate subdirectory, so that a user fetching only particular file types should find them contiguous on a tape archive. And within each of these subdirectories, the files are split into chunks called *superslabs*, as they are the concatenation of several simulation slabs. The intention of the superslab division is to allow the user to segment their loading of the simulation for a rolling processing. For the base simulations, there are 34 superslabs, each comprised of 50 slabs (with the last being 51), so each superslab is a 3 per cent planar region of the survey volume. We also provide the parameter file for the simulation invocation, the linear power spectrum file, and a summary file that contains a table of cosmological epoch, timing, and statistics for each time-step.

We use ASDF (Advanced Scientific Data Format, Greenfield, Droettboom & Bray 2015) as our container file format, as this offers

Table 4. Data products available for an output redshift of AbacusSummit, as part of the CompaSO halo catalogue.

`uint64_t id`	A unique halo number.
`uint64_t npstartA`	Where to start counting in the particle output for subsample A.
`uint64_t npstartB`	Where to start counting in the particle output for subsample B.
`uint32_t npoutA`	Number of taggable particles written out in subsample A.
`uint32_t npoutB`	Number of taggable particles written out in subsample B.
`uint32_t ntaggedA`	Number of tagged particle PIDs written out in subsample A.
`uint32_t ntaggedB`	Number of tagged particle PIDs written out in subsample B.
`uint32_t N`	The number of particles in this halo.
`uint32_t L2_N[N_LARGEST_SUBHALOS]`	The number of particles in the largest L2 subhaloes.
`uint32_t L0_N`	The number of particles in the L0 parent group.
`float SO_central_particle[3]`	Coordinates of the SO central particle.
`float SO_central_density`	Density of the SO central particle.
`float SO_radius`	Radius of SO halo (distance to particle furthest from central particle).
`float SO_L2max_central_particle[3]`	Coordinates of the SO central particle for the largest L2 subhalo.
`float SO_L2max_central_density`	Density of the SO central particle of the largest L2 subhalo.
`float SO_L2max_radius`	Radius of SO halo (distance to particle furthest from central particle) for the largest L2 subhalo.
The quantities below are computed using a centre defined by the centre of mass position and velocity of the largest L2 subhalo. In addition, the same quantities with `_com` use a centre defined by the centre of mass position and velocity of the full L1 halo. All second moments and mean speeds are computed only using particles in the inner 90 per cent of the mass relative to this centre.	
`float x_L2com[3]`	Centre of mass position of the largest L2 subhalo.
`float v_L2com[3]`	Centre of mass velocity of the largest L2 subhalo.
`float sigmav3d_L2com`	The 3D velocity dispersion, i.e. the square root of the sum of eigenvalues of the second moment tensor of the velocities relative to the centre of mass.
`float meanSpeed_L2com`	Mean speed of particles, relative to the centre of mass.
`float sigmav3d_r50_L2com`	Velocity dispersion (3D) of the inner 50 per cent of particles.
`float meanSpeed_r50_L2com`	Mean speed of the inner 50 per cent of particles.
`float r100_L2com`	Radius of 100 per cent of mass, relative to L2 centre.
`float vcirc_max_L2com`	Max circular velocity, relative to the centre of mass position and velocity, based on the particles in this L1 halo.
`int16_t sigmavMin_to_sigmav3d_L2com`	Min(`sigmav_eigenvalue`) / `sigmav3d`, condensed to [0,30000].
`int16_t sigmavMax_to_sigmav3d_L2com`	Max(`sigmav_eigenvalue`) / `sigmav3d`, condensed to [0,30000].
`uint16_t sigmav_eigenvecs_L2com`	Eigenvectors of the velocity dispersion tensor, condensed into 16 bits.
`int16_t sigmavrad_to_sigmav3d_L2com`	`sigmav_rad` / `sigmav3d`, condensed to [0,30000].
`int16_t sigmavtan_to_sigmav3d_L2com`	`sigmav_tan` / `sigmav3d`, condensed to [0,30000].
`int16_t r{10,25,33,50,67,75,90, 95,98}_L2com`	Radii of this percentage of mass, relative to L2 centre. Expressed as ratios of `r100` and condensed to [0,30000].
`int16_t sigmar_L2com[3]`	The square root of eigenvalues of the moment of inertia tensor, as ratios to `r100`, condensed to [0,30000].
`int16_t sigman_L2com[3]`	The square root of eigenvalues of the weighted moment of inertia tensor, in which we have computed the mean square of the normal vector between the COM and each particle, condensed to [0,30000].
`uint16_t sigmar_eigenvecs_L2com`	The eigenvectors of the inertia tensor, condensed into 16 bits.
`uint16_t sigman_eigenvecs_L2com`	The eigenvectors of the weighted inertia tensor, condensed into 16 bits.
`int16_t rvcirc_max_L2com`	radius of max circular velocity, relative to the L2 centre, stored as the ratio to `r100` condensed to [0,30000].

human-readable headers while allowing us to separate each column of our catalogues into a separate block. This allows users to efficiently load subsets of the columns, and we chose an ordering of columns that we hope will yield some contiguous loading patterns. Further, we chose the size of the superslabs so that the data volumes in individual columns in these files would typically be at least 10 MB, so as to amortize disk seek-time latency.

For the particle subsamples, as many applications only need particles in the haloes, we separate the particles into two sets of files for halo and field, based on the L0 segmentation. It is important to note that while the chunk boundaries of the field file are set simply by the cell boundary, i.e. clean planar cuts, the boundaries of the halo file are set based on whether the halo (not the particle!) belongs to a cell. Haloes frequently span multiple cells, so the halo files do not have a clean planar boundary. To get a complete union of particles in a small region, one may need to draw from two consecutive halo-set files. We further separate the halo and field files into a 3 per cent and 7 per cent subsample, called A and B, so that users can efficiently access 3 per cent, 7 per cent, or 10 per cent of the particles for their application.

We were aggressive in trimming the bit-level precision of output quantities, as the low-significance bits are of little to no science value but foil data compression algorithms. Notably for the particle subsamples, we limited positions to 20 bits across the box (about 2 kpc h^{-1} granularity and 0.6 kpc h^{-1} rms) and velocities to 12 bits from -6000 to $+6000$ km s^{-1} (3 km s^{-1} granularity and 0.9 km s^{-1} rms); these were combined into a single 32-bit integer for simple file loading, and therefore we refer to the format as `rvint`. For the full particle time slices, where there are many particles in a single cell, we have a more complicated format called `pack9` in which positions and velocities are stored with 12 bits each, but relative to a scaling in a per-cell header. This gives mildly higher resolution

(0.3 kpc h^{-1} granularity) while holding the compressed size to 8.5 bytes per particle.

Particle ID numbers and the kernel density estimates are placed in separate order-matched files, using a 64-bit integer. We store the square-root of the density estimate in units of the cosmic mean as an integer. We note that the small radius of the kernel is such that even a single particle gives a density of about 10 in these units; i.e. the density is useful in halo work, but is not a good measure of densities near the cosmic mean.

Halo catalogue columns were carefully considered for suppression of bit precision. For example, radii are reported as a ratio to the maximum particle radius in the halo, because this ratio falls in the range 0 to 1, we compressed it to a 16-bit unsigned integer. Similar ratios are used for velocity dispersions relative to the three-dimensional dispersion, which provides a robust maximum. The rank-3 moment of inertia tensors were diagonalized so that the eigenvalues could be outputted as ratios and the three Euler angles of the eigenvectors compressed heavily into a single 16-bit integer giving few degree precision. We provide a PYTHON reader, distributed on GitHub and PyPI as `abacusutils`,[2] that can read subsets of columns and that performs a translation of all of these ratios and other compressions back into physical units.

After this explicit truncation of precision, we then perform lossless data compression with the `zstd`[3] algorithm in the `Blosc` package.[4] Prior to compression, we perform transposition of the bits or bytes in the data vector, a feature built into `Blosc`. In other words, if one has 1 million 4-byte numbers, byte transposition would yield a vector of the million first bytes, followed by the million second bytes, etc. Bit transposition would have the million first bits, then the million second bits, etc. This helps the compression ratios substantially, and we recommend investigating this in any data compression of a vector of fixed-sized numbers.

We were disappointed at how poorly standard algorithms perform at identifying patterns that occur every 4th byte, as commonly happens with integer data that only rarely produce values above 2^{16} (but can and therefore can't be shorted to 16 bits) or with floating-point data whose exponents do not explore a wide range. After bit or byte transpose, these patterns are made contiguous in the file and the compression algorithms squeeze the result to almost nothing. For example, for our storage of 4-byte healpix pixel numbers (which reach 3 billion), we sorted the values in internally convenient segments of the output and then did this compression; the result requires only one-third of the byte per particle.

Rather than transposing an entire column, higher performance can be gained by performing the transposition in cache-sized blocks. The default `Blosc` recommendation is to target the CPU's L1 cache size (typically 32 KB), and while this yields high performance, we found that this small size missed substantial compression opportunities. We studied the performance/compression trade-off of various block sizes, and found that a few MB yielded nearly maximal compression while still providing >500 MB s^{-1} of decompression per core, including both the inverse transpose and the `zstd` decompression. Since the decompression parallelizes over blocks, this was deemed fast enough to keep up with almost any storage medium with only a few cores.

Similarly, we experimented with several backing compressions before settling on `zstd`. `lz4` offered a compelling alternative, and indeed was faster than `zstd`, but missed substantial compression opportunities that `zstd` found even at its lowest compression level. The resulting smaller file sizes also result in higher performance in the limit of slow storage media.

We provide an extension to the ASDF PYTHON reader that incorporates the `Blosc` engine, so that all of the decompression (including the inverse transpose) is invisible to the user. The extension is transparently installed via the ASDF compression extension mechanism, which was developed in support of this project, when the user installs the `abacusutils` PYTHON package.

For base simulations without full time-slice outputs, the total set of data products is about 7.85 TB. This is about 2.2 TB for the 33 redshift catalogues, 1.3 TB for the light cone information, and 4.4 TB for the particle subsamples. When time-slice outputs do exist, they are 3.5 TB per redshift, which is about 10.6 bytes per particle including the particle ID.

All ABACUS data products undergo 32-bit checksum verification to ensure accuracy and guard against corruption as they flow through the ABACUSSUMMIT production pipeline, from production to compression all the way through to data transfer to their final location. We have implemented `fast-cksum`[5]: a checksum utility that produces identical output to the GNU `cksum` utility provided in most Linux environments, but with roughly a factor of 10 improvement in speed. The performance is obtained from pre-computing look-up tables for all possible 16-byte values convolved with the CRC generating polynomial.[6]

Using `fast-cksum`, ABACUS computes and stores the checksums of all data products on-the-fly before the data is written out to disc. Every subsequent time the data products are read off of disc during post-processing, the ABACUSSUMMIT production pipeline recomputes each file's checksum and verifies it against the recorded original checksum before proceeding to compression. After data compression, we store a new list of checksums for all compressed data product files and verify these, again, during data transfer to NERSC and OLCF HPSS.

3 ABACUS

3.1 Force solver algorithm and serial method

The ABACUSSUMMIT suite was executed on the Summit supercomputer using the high-performance, high-accuracy *N*-body code ABACUS. ABACUS uses a variant of the Fast Multipole Method first developed in Greengard & Rokhlin (1987) to calculate gravitational forces. ABACUS' force-solver algorithm, introduced in Metchnik (2009) and detailed in Pinto et al. (in preparation), is built on the structure of the simulation space: the total simulation volume consists of a single, cubic box containing *N* particles, and an infinite number of its periodic replicas. Each box is, in turn, subdivided further into a cubic lattice of cells (with $\texttt{CPD}^3$ cells, where `CPD` stands for 'cells per dimension'). Each two-dimensional slice of the box, containing $\texttt{CPD}^2$ cells, is called a slab.

To solve for the gravitational force on a single particle, ABACUS decomposes the force into a 'near field' – the force arising from nearby cells – and a 'far field' – that arising from well-separated cells. This division is set by the *near-field radius*, called *R*. The near- and far-field terms are computed independently using separate techniques to balance accuracy with computational cost. Because the

[2] https://github.com/abacusorg/abacusutils
[3] https://facebook.github.io/zstd/
[4] https://www.blosc.org/
[5] https://github.com/abacusorg/fast-cksum/
[6] This approach was derived from https://github.com/stbrumme/crc32

separation between the near and far fields is disjoint, every pairwise interaction is present in only one or the other. We note that, formally, every pair of particles has an infinite number of pairwise interactions – one for each periodic image – and that distant periodic replicas of the near-field particles are included in the far-field force computation. There is no approximation of close particle pairs as non-periodic.

The far-field acceleration is represented by Taylor-series expansions, up to order p, in each cell. The coefficients of these expansions are computed by cyclically convolving (in Fourier space) the multipole moments of the mass distributions in the cells with a kernel consisting of derivatives of the Green's function, which we refer to as the 'derivatives tensor'. The derivatives tensor depends only on the geometry of the grid and not on the cell's contents and may therefore be pre-computed at the beginning of the simulation and re-used at every time-step. Furthermore, both the multipole moments and the derivatives tensor possess many simplifying symmetries and recursion relationships which significantly reduce the complexity of the calculation. The multipole order parameter p sets the accuracy of the Taylor approximation and the desired choice of p depends on the distance between the two domains; in our case, $p = 8$ yields excellent force accuracy while maintaining high performance given a separation of two cells ($R = 2$).

The near-field force on a given particle is sourced by particles in neighbouring cells out to radius R and can be computed by any open-boundary condition Newtonian force solver; in the simulations presented, it is computed by a direct, $O(N^2)$ summation with a compact spline softening kernel. Direct summation proves to be well suited to GPUs, as the highly regular geometry of the work exposes massive parallelism and efficient memory access, communication, and computation patterns.

In both the serial and parallel implementations, a single time-step of Abacus consists of two primary substeps: `singlestep` and `convolution`. We summarize the overall method in this section and describe the differences between the serial and parallel implementations of both substeps in the sections that follow.

`singlestep` flows through the grid of cells in a single direction using a 'slab pipeline', operating on a rolling subset of contiguous slabs loaded in memory. While the GPU is computing the near-field work, the CPU is actively evaluating the far-field force from the supplied Taylor series coefficients in that cell, packaging the near-field work for the GPU, co-adding the two forces, performing the leapfrog (kick-drift-kick) integration (Quinn et al. 1997), moving particles to their new cells, and computing the multipoles in these new cells. When the slab pipeline has swept through the entire box, the global step advances and the `convolution` step is invoked, producing new Taylor coefficients (see below). The near field radius R sets the minimal width of the slab pipeline: the central slab requires R loaded slabs on either side to compute the near force. In the event that Abacus needs to perform on-the-fly halo finding (described further in Section 3.3), the pipeline must broaden sufficiently to search through slabs containing particles in the same halo. Because the far-field force on a given cell is entirely represented by the Taylor coefficients, Abacus never needs to load the entirety of the particle data into memory to calculate forces. If enough memory is available, however (such as on a GPU cluster), then the full particle set may be held in memory.

During the `convolution` step, the Taylor coefficients are calculated that are required to evaluate the far-field force during `singlestep`. This is done by convolving the multipole moments of each cell computed during the previous `singlestep` with the pre-computed derivatives tensor to obtain the Taylor coefficients representing the approximation of the far-field potential at the centre of each cell. These may be saved to disk at the conclusion of `convolution`, or held in memory if desired. The convolution can be performed in Fourier space as a multiplication, so we take the YZ-FFT during `singlestep` while the YZ slab is in memory. The convolution then performs the cross-slab X-FFT, multiplies it using the convolution theorem with the Fourier transform of the derivatives tensor, and concludes by applying the inverse X-FFT to obtain the Taylor coefficients. The inverse YZ-FFT is completed during `singlestep` immediately before applying the far-field force to a slab.

The slab pipeline makes it simple to balance CPU and GPU loads, and to mask the I/O (performed on a dedicated thread) with the bulk of the CPU work. The slab pipeline also affords Abacus a unique opportunity: since only a small fraction of the simulation volume need be held in memory at any given time, Abacus can run, on a single node, simulations that would force another N-body code to seek an allocation on a large computer cluster. However, for AbacusSummit, we ran entirely in-memory, since we had available the memory-rich Summit cluster.

3.2 Parallel method

In its single-node implementation, Abacus' algorithm enables the execution of unprecedentedly large simulations without holding the whole state in memory. This approach naturally lends itself to multi-node parallelization, since, to a given node, it does not matter if the not-in-memory state is on disc or resident in another node's memory. This is how AbacusSummit was run, with the simulation state residing entirely in memory, distributed over nodes, using the slab pipeline to organize the computation.

The parallel domain decomposition reuses the slab decomposition of the simulation volume, with each node responsible for a contiguous span of slabs at each time-step. A given node begins a time-step with slabs j through $j + N$ in memory, with slabs $j - N$ through $j - 1$ on the 'left' neighbour node, and slabs $j + N$ to $j + 2N$ on the 'right' neighbour node. The node processes its slabs sequentially, with slab $j + 2$ being the first to receive forces (due to the near-field radius), and slab k being the first to finish. On a non-group-finding step, $k = j + 3$, since particles from neighbouring slabs must be allowed to drift into this slab, but during group-finding epochs, k may be $j + 10$ or greater.

Having completed all work for slab k, the slabs below k require information from slabs below j, which are not in memory because they reside on the left neighbour node. The slabs below k, however, do not require any more information from slabs k and above. Therefore, the node can 'detach' slabs $k - 1$ and below and send them to its left neighbour node, where they will eventually be completed. The transfer is accomplished with an asynchronous MPI send.

Because a node begins the time-step with slabs j to $j + N$ in memory but ends with slabs k to $k + N$ in memory, the domain decomposition is said to rotate over the nodes. A given node is therefore not responsible for a static portion of the simulation volume, but a cyclically rotating slice. Information is only sent once per time-step and the network load is easily overlapped with other computation, since it need only arrive before the node's work has reached the other side of its domain. There is no need for nodes to frequently synchronize over stages of work, as would happen with a 2D or 3D decomposition.

In the serial implementation, at the end of `singlestep` all the YZ slabs' multipole moments are stored on a single node and are available to the following `convolution`. Subdividing the slab domain amongst the compute nodes in the parallel implementation

leaves each node with a finite range in x of YZ slabs' multipole moments stored in its memory. Recall that `singlestep` has already performed the YZ-FFT; `convolution` is responsible for performing the cross-slab forwards and inverse X-FFTs. For this, each node requires the full range of x of multipole moments slabs, but does not require the full range of z – the X-FFTs can be performed on chunks of arbitrary thickness in the z direction. Thus, to parallelize the `convolution` module, we begin with an MPI `Alltoall` communication between the compute nodes: each node starts with several contiguous YZ slabs of multipole moments, and ends up with several contiguous XY slabs of multipole moments held in its memory. Each node then performs the X-FFT to complete the transformation of the multipole moments to Fourier space, multiplies them using the convolution theorem with the derivatives tensor, and takes the inverse X-FFT to obtain the Taylor coefficients. The `convolution` module concludes with an inverse `Alltoall` that stores, on each node, the Taylor coefficients belonging to the node's original YZ slab domain, leaving the nodes prepared to proceed to `singlestep`.

The simplicity of the 1D domain decomposition comes at a price: the nodes need to have enough memory and enough compute power to service a slice of the box sufficient to hold the full width of the pipeline. In ABACUS simulations, this width is set by the need to identify dark matter haloes on-the-fly, up to a natural diameter of about $10\,h^{-1}$ Mpc. This in turn requires a computational domain of 25–30 h^{-1} Mpc thickness, which contains about 5 billion particles. Summit's 512 GB of RAM per node enabled ABACUS to store a wide-enough slice to adopt this 1D parallelization scheme for the production of ABACUSSUMMIT.

The serial version of ABACUS has been extensively validated and compared to other N-body codes, as described in Garrison et al. (2018a,b, 2021a). For given initial conditions, ABACUS generates results that are highly reproducible in the low-redshift summary statistics. At all stages of code development, we were therefore able to confirm that the parallel code base agrees to nearly machine precision in the full-simulation summary statistics, and on a particle-by-particle basis for single time-steps.

3.3 Halo finding

ABACUS identifies haloes with COMPASO: a new, hybrid, on-the-fly halo finder described in Hadzhiyska et al. (2021). Performing halo finding on-the-fly avoids the need to write and retrieve the full particle data for post-processing. ABACUS uses the common friends-of-friends method (Davis et al. 1985) to partition the particles into strictly disjoint sets. Within each set, we then use COMPASO to perform a spherical overdensity (SO) method to identify science haloes (Lacey & Cole 1994; Tinker et al. 2008). We then perform a second round of subhalo finding with a higher density contrast. These are used as centres for halo moments, but are not reported as catalogue entries in their own right.

In detail, halo finding begins by computing a kernel density estimate around all particles, using a weighting of $1 - r^2/b^2$, where b is chosen to be 40 per cent of the interparticle spacing. This step occurs as part of the near-field force computation with minimal computational burden, and we include the resulting density in our data products for use in other analyses. The particle set is then segmented into so-called level zero (L0) haloes using the friends-of-friends (FOF) algorithm with linking length set to $b_{\rm FOF} = 0.25$ of the interparticle spacing, but including only particles with a relative overdensity contrast $\Delta = 60$. We note that the percolation density for $b_{\rm FOF} = 0.25$ is 41.8 (Lorenz & Ziff 2001), well below 60. The intention is that the bounds of the L0 haloes be set by the kernel density estimate, which has lower variance than the nearest neighbour method of FOF and imposes a physical smoothing scale.

Here and below, the density thresholds are scaled upward as the cosmology departs from Einstein-de Sitter, in keeping with spherical collapse estimates for low-density universes. We define Δ values relative to critical density scaled by the Bryan & Norman (1998) factor of $(1 + 82x - 39x^2)/18\pi^2$, where $x = \Omega_M(z) - 1$. The FOF linking length is scaled as the inverse cube root of that change, while the kernel density scale b is not changed.

All subsequent L1/L2 group finding and all halo statistics included in ABACUSSUMMIT are based solely on the particles in their L0 halo. Within each L0 halo, COMPASO constructs L1 haloes by a competitive spherical overdensity algorithm. The particle with the highest kernel density is selected to be the nucleus of a new halo, and COMPASO then searches outward to find the innermost radius in which the enclosed density of L0 particles crosses below the chosen $\Delta = 200$ threshold. Particles within this radius (labelled R_{200}) are tentatively assigned to the L1 group, and particles within 80 per cent of R_{200} are marked as ineligible to be a future nucleus. Among the remaining eligible particles, COMPASO then locates the particle with the highest kernel density that is also denser than all other particles (eligible or not) within 40 per cent of the interparticle spacing. If the located particle has a density higher than the density generated by a singular isothermal sphere with 35 particles within R_{200}, the located particle is designated as a new L1 nucleus. For each new L1 nucleus, COMPASO searches in the set of all L0 particles for the R_{200} SO radius. Each L0 particles is assigned to the new L1 group if it had previously been unassigned or if it is estimated to have an enclosed density with respect to the centre of the new group that is twice that of the enclosed density with respect to its assigned group. The enclosed densities are estimated using an inverse square density profile scaled from the SO radius. We note that COMPASO does not perform any unbinding of particles based on their gravitational potential, as in the case of other halo finders, e.g. ROCKSTAR (Behroozi, Wechsler & Wu 2012).

Within each L1 halo and using only the L1 particles, COMPASO repeats the SO search to locate L2 haloes with an enclosed density contrast of 800. ABACUSSUMMIT uses the centre of mass of the largest L2 subhalo to define a centre for computing L1 halo statistics. ABACUSSUMMIT further outputs the masses of the five largest L2 subhaloes; however, no further information about the L2 substructure is included in the ABACUSSUMMIT data products.

COMPASO has been compared extensively and compared against existing halo finders; the results are described in Hadzhiyska et al. (2021).

3.4 Merger trees and cleaned halo catalogues

Following the identification of spherical overdensity haloes using the COMPASO algorithm, we construct halo merger trees as a post-processing step using the 33 halo output epochs. Halo merger trees correspond to associations between L1 haloes identified across output times, resulting in lists of progenitor and descendant haloes. Our merger tree algorithm works in a *reverse time order*, i.e. traversing the halo catalogues from low to high redshift.

To accelerate the process of associating a halo, `halo_now`, at snapshot `i`, with its progenitors in snapshot `i-1`, we 'pre-filter' the COMPASO catalogue by first selecting only those haloes at snapshot `i-1` that are within at most a distance of 4 Mpc h^{-1} from `halo_now`. This narrows down the search to a much smaller list of plausible halo associations.

From this list, we then identify *candidate associations* as those haloes that donate a non-zero fraction of their unique particle IDs, $f_{\rm donate}$, to `halo_now`. We also record the fraction of subsampled particles in `halo_now` that are donated to it by its candidate associations as $f_{\rm match}$. Note that all matched fractions are weighted by the particle kernel density, which gives preferences to associations that donate particles to the central core of `halo_now`. Finally, we mark a candidate association as a `Progenitor` if $f_{\rm donate} \geq 0.5$. Furthermore, the `MainProgenitor` is identified as the association that contributes the largest $f_{\rm match}$. We repeat this procedure for all AbacusSummit haloes containing at least 50 particles, and with at least five subsample particles available for tracking.

A primary application of our merger trees is to post-process and 'clean' the raw CompaSO halo catalogues output by Abacus. This procedure involves identifying haloes with non-monotonic mass growth as a result of one or more dynamical events in their past histories including fly-bys, halo 'splits' (where a single halo may be deblended by CompaSO), partial mergers, etc. We identify these objects as those whose peak mass is at least two times greater than their present-day mass. We then traverse the merger tree of each flagged object to find a nearby, neighbouring halo that shares the same progenitors as the flagged halo; the two haloes are then merged at that redshift and remain merged at all subsequent times. The net effect is to 'clean' the CompaSO catalogues of several low-mass haloes, typically found at the peripheries of larger systems. The masses of these larger haloes are then incremented by the mass of the individual haloes merged on to them. The full details of the cleaning algorithm are described in Bose et al. (2021), where we also demonstrate the various benefits of cleaning the CompaSO catalogues using this method.

3.5 Light cones

AbacusSummit simulations produce a light cone centred at the corner of the box and include one periodic copy of the box displaced in the y-direction, and one displaced in the z-direction. At every time-step, Abacus identifies particles that belong to both the light cone and the 10 per cent particle subsample (the concatenation of the A and B subsamples; see Section 2.6), and outputs their positions, velocities, particle IDs, and HEALPix pixel number that can be used to form projected density maps. The HEALPix pixels are computed using $+z$ as the North Pole, i.e. the usual (x, y, z) coordinate system, with the Nested pixel labelling.

For base boxes with length 2000 h^{-1} Mpc on a side, we position the light cone observer at (–990, –990, –990), or, in other words, 10 h^{-1} Mpc inside a box corner. Abacus sweeps through the box and two of its periodic copies – these three boxes forming the eligible space of the light cone are centred at (0,0,0), (0,0,2000), and (0,2000,0), respectively (where all lengths are given in h^{-1} Mpc units). This provides an octant to a distance of 1990 h^{-1} Mpc ($z = 0.8$), shrinking to two patches each about 900 square degrees at a distance of 3990 h^{-1} Mpc ($z = 2.45$).

The three copies of the box are output separately and the particle positions are referred to the centre of each periodic copy: e.g. the particles from the higher redshift box need to have 2000 h^{-1} Mpc added to their z coordinate.

For the two `huge` boxes of 7.5 h^{-1} Gpc, we position the light cone observer at the centre of the box (0,0,0). This provides a full-scale light cone to 3750 h^{-1} Mpc distance ($z = 2.18$), with smaller areas reaching to 6495 h^{-1} Mpc in the 8 corners. For example, half the sphere is available to 4500 h^{-1} Mpc ($z = 3.2$).

The algorithm for identifying particles belonging to the light cone is as follows. At each timestep, Abacus computes the radii of the light cone at the beginning and at the end of the step. The goal is to identify every particle which, during the upcoming time-step, will pass through the light cone. To do this, we check each cell in the simulation box to determine whether its centre is in the vicinity of the light cone. If not, the cell may be skipped. If the cell is close to the light cone, however, Abacus 'opens' the cells and considers each particle in the given cell individually.

Using velocity-extrapolated leapfrog integration, Abacus computes the fraction of the time-step (`fracstep`) when a given particle is expected to meet the light cone sweeping through its host cell (accounting for periodic boundary wraps as needed). If `fracstep` is between zero and one, this implies that the particle will cross the light cone during the given time-step and therefore should be flagged for output. The particle's position and velocity are extrapolated using drifts and kicks, respectively, to the time at which it will cross the light cone. To guard against floating point errors arising from subtracting two numbers close in magnitude when computing `fracstep`, we accept values of `fracstep` just below zero (in the case of AbacusSummit, we set the tolerance to 10^{-6}), so that particles that fall just behind the light cone during the time-step are nevertheless include in the output.

We note that we do not perform extrapolation across the transverse boundaries of the light cone (e.g. the boundaries bordering the edge of one of the three periodic copies of the box that is not adjacent to another copy). Therefore, the light cones are accurate only out to a distance corresponding, conservatively, to the width of one Abacus cell (`BoxSize/CPD`) from their transverse edges. In the case of most AbacusSummit simulations, this distance is equal to approximately 1.2 h^{-1} Mpc. However, this estimate is conservative; a particle typically moves much less than the width of a cell per time-step, so it is therefore highly unlikely that a particle $1.2h^{-1}$ Mpc away from the transverse edge would move sufficiently far such that the lack of the periodic wrap would fail to account for it.

4 PERFORMANCE ON SUMMIT

4.1 Overview

Abacus on Summit is very fast. We achieve roughly 70M particles per second per node at early times, until $z \sim 1$. Past that time, the computation becomes dominated by the near-force calculation on the GPUs, with the performance falling to 45M particles per second per node at $z = 0.1$. For comparison, a high-performing example of a previously published N-body speed was 3.8M particle-steps per second per node on Titan with PKDGRAV3 (Potter, Stadel & Teyssier 2017), although this result was on older hardware and we expect improved results could be presented by those authors.

Abacus owes its high performance to its novel force solver, which enables Abacus to run using the slab pipeline structure described in Section 3, dispatching significant work to both the CPU and GPU and overlapping their computation. In addition, we tune for performance with hardware-specific optimizations. In this section, we describe Summit's system and hardware, the achieved performance of Abacus on Summit, the optimizations and features that contribute to Abacus' speed in general and on Summit specifically, and, in brief, the performance of ancillary code modules used in the AbacusSummit production pipeline.

Table 5. Computational resources used in the production of AbacusSummit. `highbase` used 15 nodes for two sims, and 35 for one. `fixedbase` used 30 nodes for five sims, and 15 for one. Of the `small` boxes, 1643 reached the final redshift of $z = 0.2$; 1883 reached $z = 1.1$. Only the former are presented in this table to aid interpretation of the timings.

Spec.	N	Size (Mpc h^{-1})	Final z	Number of boxes	Nodes	Total node-hours (per sim)	Wall-clock (per sim)	Data products, compressed
`base`	6912^3	2000	0.1	139	60	250 K (1800)	30 h	1600 TB
`huge`	8640^3	7500	0.1	2	141	6.5 K (3300)	23 h	100 TB
`high`	6300^3	1000	0.1	1	46	1.9 K (1900)	41 h	40 TB
`hugebase`	2304^3	2000	0.1	25	5	1.1 K (43)	8.6 h	22 TB
`highbase`	3456^3	1000	0.1	2/1	15/35	790 K (260)	16/8.7 h	12 TB
`fixedbase`	4096^3	1185	0.1	5/1	30/15	2.6 K (440)	15/14 h	39 TB
`small`	1728^3	500	0.2	1643	1	32 K (20)	20 h	140 TB
Total	57 T	2400 (Gpc/h)^3	–	–	–	290 K	–	2000 TB

4.2 The summit system

Summit is comprised of 4608 IBM AC922 compute nodes, each with two 22-core IBM POWER9 processors and six NVIDIA Tesla V100 GPUs. Each compute node contains 512 GB of RAM and 96 GB of High Bandwidth Memory accessible by the GPU accelerators. The compute nodes each provide a theoretical double-precision capability of 40 TF (ORNL 2020).

Each IBM POWER9 processor utilizes IBM's SIMD Multi-Core technology (SMC); SMCs support simultaneous multi-threading (SMT) up to level 4, such that each physical core is capable of running a maximum of 4 hardware threads (compare with Intel's hyper-threading). The POWER9 processor contains 22 SMCs. To maximize our performance on the POWER9 processors, we have added VSX intrinsic functionality to relevant code modules within Abacus (Section 4.4), augmenting our existing Intel AVX capability.

The Summit node-hour usage is presented in Table 5, organized by simulation specification (`base`, `hugebase`, etc.; see Section 2.3).

4.3 Performance

The main Abacus code consists of two executables invoked in a tick-tock fashion: `singlestep` and `convolution`. `singlestep` dominates the runtime – it computes forces and performs the particle update cycle, while `convolution` only operates on the cell multipoles. Table 6 presents timings for these two executables for a typical AbacusSummit box ($N_P = 6912^3$, `BoxSize` $= 2\ h^{-1}$ Gpc) executed on 60 compute nodes. Timings at two epochs are shown $z = 90.5$ (nearly unclustered), and $z = 0.105$ (highly clustered). The total step time is 79.5 s (69.2 million particles per second) at the former epoch, and 122 s (45.1 million particles per second) at the latter. Fig. 4 shows the evolution of the runtime throughout the course of the simulation.

`convolution` takes 10–15 per cent of each step's runtime. The workload of `convolution` is independent of epoch, consistently requiring approximately 12 s for a typical AbacusSummit box. Of this time, about a third of the work occurs in executing fast Fourier transforms before performing the arithmetic required to apply the Convolution Theorem, which in turn accounts for about 40 per cent of `convolution`'s runtime. An additional 2 s are required to re-order (swizzle) arrays before and after taking their transforms, with an additional 1 s to read the lightweight Derivatives files. The GPUs are not used in this part of the code.

`singlestep` accounts for 85–90 per cent of an Abacus timestep's runtime. In this portion of the code, Abacus sweeps through the particles slab-wise, executing a series of pipeline actions on each slab. The CPU loads the particle positions, constructs the indexing structures for the GPU near-field work, computes the far-field force by evaluating the Taylor series of the potential, kicks the particle velocities, drifts the positions, and reorders particles into their new cells. Finally, the cell multipoles of the new particle positions are computed. The GPU computes near-field forces while the CPU is executing the rest of this work.

As described in Section 2, we choose the code parameter `CPD` (cells per dimension) such that the CPU and GPU workloads are balanced. Since the GPU work grows as the simulation becomes more clustered, we cannot balance the work for all time steps, but we can choose a value that minimizes the time-to-solution. For the fiducial $N_P = 6912^3$ AbacusSummit simulations, this optimum is around `CPD` $= 1701$, or 67 particles per cell.

With this `CPD`, the CPU work masks the GPU computation prior to $z \sim 1$. In this early regime, `singlestep` takes about 67 seconds (83 million particles per second) and the CPU work accounts for 63 of those, with the remainder being attributed to fixed start-up and tear-down costs (most from MPI, CUDA, and load balancing). The GPU near-field work takes only 39 seconds and is thus completely masked.

After $z \sim 1$, the near-field work dominates the runtime, reaching 97 seconds by the final redshift $z = 0.1$. `singlestep` takes 109 s in total at this epoch; the 12 s differential arises because the final kick, drift, MPI send/receive, merge, and multipoles cannot happen until the final GPU forces are computed, in addition to the usual fixed start-up and tear-down costs.

Considering the `singlestep` CPU work (59–63 s), almost half of the time is spent in two stages: the Taylor Force (15–17 s) and the Multipoles (13–14 s). The former evaluates the gradient of the Taylor series of the potential, producing the far-field force, and the latter evaluates the cell-wise multipoles for use in the convolution. Both stages include a 2D (slab-wise) FFT. These stages contain most of the floating-point CPU work and benefit from SIMD acceleration. The remaining half of the CPU time is spent in memory-bound work, like the Kick, Drift, Transpose, and Merge.

On Summit, Abacus has achieved a top GPU performance of 2200G direct pairwise force calculations per second (Fig. 5), including host-GPU communication, notionally about 56 TFlops of compute speed and 60 per cent of peak theoretical performance. The peak theoretical performance is computed by scoring each arithmetic operation in the direct kernel as 1 FLOP, and the inverse square root as 2 FLOPs, following the guidance of the NVIDIA Nsight documentation.[7] This yields 26 FLOPs per interaction. The six V100

[7] https://docs.nvidia.com/gameworks/content/developertools/desktop/analysis/report/cudaexperiments/kernellevel/achievedflops.htm

Table 6. Wall clock timing for a typical AbacusSummit time-step, from the `AbacusSummit_base_c000_ph000` box realization ($N_P = 6912^3$, `BoxSize` $= 2\ h^{-1}$ Gpc) executed on 60 compute nodes. Timings are shown for two representative steps, one at $z = 90.5$ and the second at $z = 0.105$ (*Group Finding* from $z = 0.1$ also shown). Units of Mp/s denote millions of particles per second. *Non-blocking* means other CPU actions can proceed while that action is running. *Unaccounted* time is time spent checking preconditions, or other asynchronous overheads.

Action	High-z Time (s)	Rate	Low-z Time (s)	Rate
Total	79.5	69.2 Mp/s	122.0	45.1 Mp/s
Convolution	11.7		11.8	
Fourier transforms	3.6		3.6	
Convolution arithmetic	5.0		5.0	
Array swizzle	2.1		2.1	
Disk I/O	0.9		1.1	
Single step	66.6	82.6 Mp/s	108.8	50.6 Mp/s
CPU work	63.1		58.8	
Transpose positions	4.4		4.6	
Index near force pencils	2.4		2.4	
Taylor Force	17.0	324 Mp/s	15.6	353 Mp/s
Kick	4.6		3.9	
Drift	3.2		2.9	
Finish	22.7		22.6	
Merge new particles	2.4		2.3	
Compute multipoles	13.7	401 Mp/s	12.9	427 Mp/s
∗Group finding	.		193.1	28.5 Mp/s
Unaccounted	8.8		6.8	
GPU near force (non-blocking)	39.4	140 Mp/s	96.7	56.9 Mp/s
Waiting for GPU, MPI, or I/O	2.4		44.9	

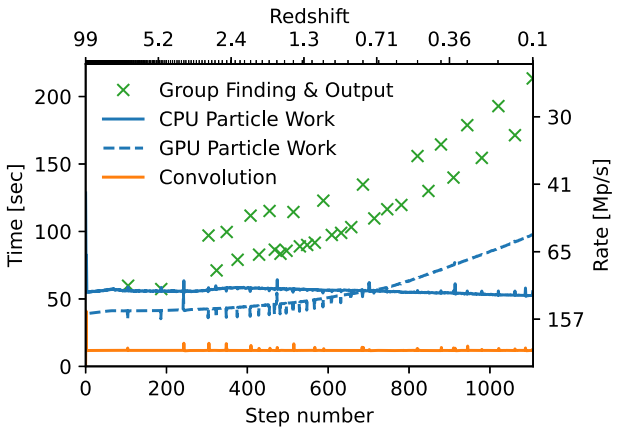

Figure 4. A timing overview of a single simulation (`base_c000_ph000`). CPU work (solid blue line) includes Kick, Drift, Multipoles, etc. (Table 6) and is approximately constant; GPU near-field force time (dashed blue line) increases as the clustering in the simulation increases. The CPU and GPU work run concurrently. The Convolution (solid orange line) precedes the CPU/GPU work at each time-step. Group-finding occurs at 33 epochs, 12 of which include full particle outputs.

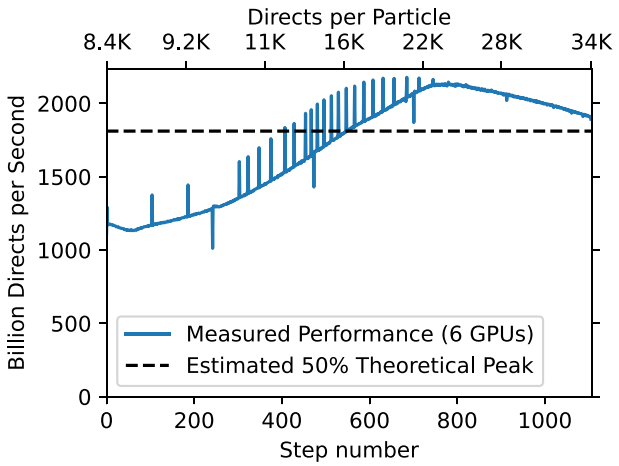

Figure 5. The GPU performance (mean across nodes) for the `base_c000_ph000` simulation, in billions of pairwise interactions (directs) per second, including overheads like host-device communication. The theoretical maximum rate is computed assuming 26 floating point operations per interaction (see the text), and assuming all of those operations can be executed with FMA – a highly conservative approximation. Under these assumptions, we observe a peak of 60 per cent of theoretical performance.

GPUs are assumed to be running constantly at their boost-clock performance of 15.7 TFLOPS,[8] which also assumes every operation is a 2-flop fused multiply-add (FMA). Since not all of the floating-point operations in our kernel can be expressed as FMA, we are likely close to the peak performance that can be achieved for this particular kernel implementation. The lower off-peak performance can be attributed to lower mean flop-to-byte ratio at early times, and the large number of sparse (void) kernels at late times.

The GPU throughput could have been modestly improved in several ways, but at a worse time-to-solution. Larger cells (lower `CPD`) would increase the FLOP density, but require more FLOPs overall. We additionally could have more efficiently overlapped GPU compute and communication by using three CUDA streams per GPU instead of two, but this negatively impacted CPU performance and resulted in a slower runtime. In early-time, I/O-bound steps

[8] https://images.nvidia.com/content/technologies/volta/pdf/volta-v100-datasheet-update-us-1165301-r5.pdf

where the CPU spent a large fraction of time idle, the GPU performance increased about 20 per cent, which is seen as the spikes of increased performance in Fig. 5. This indicates that host-side resource contention is likely an early-time bottleneck, and with less memory pressure during the I/O steps, the positions and accelerations can be filled and communicated more efficiently.

In `singlestep`, network loads are low; there is only one burst of transfer along the 1-d torus described in Section 3, and it takes about 5 per cent of the runtime, arriving well before it is needed. Throughout the production of AbacusSummit, we consistently saw an additional several seconds of MPI-related spinning in our timesteps. We speculate this spinning occurred as a result of delays in the compute nodes accessing memory buffers allocated internally by the MPI library. Additionally, operating system functions such as `munmap()` contributed to the runtime, albeit mostly in a non-blocking manner. In the case of `munmap()`, we devote a separate thread to freeing POSIX shared memory so that it is able to run in the background while the CPU work proceeds. It requires on the order of 25 seconds and is non-blocking, but likely contributes extra memory pressure (for more on the usage of POSIX shared memory and the associated overheads, see Garrison, Eisenstein & Maksimova 2021b).

The outer-most level of Abacus consists of a Python driver script, which loops over the timesteps and invokes the parallel job dispatcher (called `jsrun` on Summit, which uses IBM's Spectrum LSF batch scheduling system) once per timestep. In addition, the Python driver periodically checkpoints the simulation state by launching a copy from each node's POSIX shared memory to network storage. A typical checkpoint takes on the order of 110–150 seconds and runs 1-2 dozen times, depending on the simulation specifications and our optimism about the cluster state. Finally, while each `jsrun` call incurs an overhead of less than 1 second, there is some overhead to relaunching the executable and initializing libraries like CUDA and MPI for each time step. Future updates to Abacus will obviate the need to invoke a `jsrun` command multiple times per simulation.

The `base` simulation boxes required about 1100 time-steps to reach $z = 0.1$ from an initial redshift of $z_{\rm init} = 99$. A typical `base` simulation – including the Python wrapper, backups, and main Abacus execution – required 1800 node-hours to complete.

4.4 SIMD multipoles and Taylors

We optimize the CPU computation of the multipoles and Taylors using single-instruction, multiple-data (SIMD) instructions (also known as vector instructions). We use the vector extensions of the POWER9 AltiVec instruction set via the built-in VSX 'intrinsic' functions of the GCC compiler. As described in the Abacus code paper (Garrison et al. 2021a), we use a Python meta-code to manually unroll the triple-nested loop over multipole order that the multipole and Taylor computations require. The meta-code emits intrinsics to process two double-precision particle coordinates per 128-bit VSX vector, interleaving multiple VSX vectors to mask instruction latency.

We find that an eight-fold unrolling of the particle loop (i.e. interleaving 4 VSX vectors) is optimal. A POWER9 core is capable of launching 2 VSX vector operations per cycle, for a peak of 8 double-precision operations per cycle, assuming FMA.[9] With the interleaved code, we find that SMT1, 2, and 4 perform the same, while SMT 2 and 4 perform better without interleaving. This is consistent with the understanding that SMT allows the processor to mask latency by 'backfilling' with instructions from other hardware threads, but that this is not necessary if the interleaving already masks the latency.

Although the VSX vectors are relatively narrow, each Summit node has 42 user-facing cores per node, distributed over 2 sockets (1 core per socket is reserved for system use). We use 35 of these for floating-point work (or, more specifically, 70 hardware threads because other areas of the code benefit from SMT2). In production simulations, we process the multipoles kernel at about 740 million particles per second, or about 20 per core. Artificial benchmarks were only a few to 10 per cent faster per core.

The Taylors kernel computation is slower per-particle than the multipoles, but is more efficient in terms of floating-point operations per second (FLOPS), because the FLOP count per-particle for the Taylors is several times that of the multipoles. The VSX calculations all give the same result as their non-vectorized counterparts to within rounding error.

4.5 Thread scheduling and NUMA

Summit nodes employ a non-uniform memory access (NUMA) architecture, with each of its two POWER9 processors associated with one half of the system memory. Memory access within the associated half (within the NUMA node) is faster than access to the other half. Our NUMA strategy is to divide each slab between the two NUMA nodes. Within our OpenMP thread pool, each thread only works on the half of the slab belonging to its core's NUMA node. Threads are pinned to cores with the OpenMP affinity mechanism, which also ensures no costly migration of threads between cores or contention between threads. We employ a custom OpenMP scheduler, written using OpenMP Tasks and atomic addition, which allows dynamic scheduling of threads within their NUMA nodes.

For the production of AbacusSummit, we use the POWER9's simultaneous multi-threading (SMT) to use multiple hardware threads per CPU core. This improves performance of some areas of the code by allowing each core to run several instruction streams simultaneously. Of the three available SMT levels – SMT1, SMT2, SMT4 – we have found that SMT2 (two hardware threads per core) produces the best performance. SMT2 yields 84 available threads – we use 70 of those (35 physical cores) for OpenMP threads, 12 threads dedicated to interfacing with the GPUs, and 1 thread dedicated to running `munmap()` to free pages of memory, and 1 thread working on I/O.

In the computation of the near-field force, each slab is divided into spatially compact GPU work units. Work units are dispatched to GPUs via queue system, with one queue per NUMA node. Work units are dispatched to the queue of their NUMA node. Each of the 12 GPU threads (2 per GPU) listens to the queue of its NUMA node, hence work is dynamically dispatched to GPUs while maintaining NUMA locality. Host-GPU communication is done via pinned memory buffers; the initial pinning time at the beginning of each time step is a noticeable overhead but is masked by the convolution.

4.6 Production pipeline

With over 150 simulation boxes to run, automation of the steps to prepare, run, and post-process each simulation was a priority, so as to increase efficiency of production and avoid human error. These tasks were assembled in a Python pipeline that generated the simulation parameter files, queued the generation of initial conditions

[9] https://openpowerfoundation.org/?resource_lib=POWER9-processor-users-manual

on OLCF's Rhea system, executed Abacus on Summit, queued post-processing on Rhea, and finally sent the data products to OLCF HPSS (tape) and to NERSC. The pipeline was organized as a progression of 'stages', each with an 'indicator' that detected whether a particular stage had completed by examining the queue state or reading files on disk, for example. The current stage of a simulation box was compared against its last-known stage, to check for regressions or an inconsistent state. This production pipeline handled the flow of the simulations across the four systems involved in the creation of AbacusSummit: Summit, Rhea, OLCF HPSS, and NERSC.

While the execution of Abacus is by far the lengthiest step in the production pipeline, both the generation of initial conditions and the post-processing constitute important parts of the overall suite production. We summarize them below.

4.7 Initial conditions

Abacus uses `zeldovich-PLT` (Garrison et al. 2016) to generate initial conditions on ORNL's `rhea` computer. `zeldovich-PLT` is designed to run out-of-core, buffering state on disc, and is therefore capable of producing initial conditions on a single node even for very large problems. Given the availability of the Rhea cluster at OLCF and the fast Alpine network file system, we opted to use `zeldovich-PLT` out-of-core, rather than parallelize it for distributed memory systems. The performance of the code was bound by the I/O speed, but the runtime (18 node-hours) was still small compared to the time to execute a simulation, and so did not need to be optimized further.

4.8 Post-processing

During the course of a simulation, Abacus writes most data products in a raw binary format. Each simulation slab is written to a separate file, so the I/O is trivially parallelized over ranks. These raw binary formats are efficient to write, but are not optimized for space and are not self-documenting. Therefore, after reaching its terminal redshift, each AbacusSummit box is post-processed to compress the final data products and package them in a self-describing file format (ASDF, Section 2.7) before being transferred to NERSC and OLCF HPSS.

The set of post-processing tasks is quite heterogeneous, with large variations in the data volumes across redshift and, in the case of light cones, across nodes. The number of tasks could be 10s of thousands – too many to send to the batch scheduler, with a large uncertainty in the proper amount of time to request for each, especially given the reliance on network file system performance. Therefore, to mitigate the heterogeneity, we use the disBatch[10] code to do dynamic dispatch within a given resource allocation. The individual tasks are itemized in a 'task file', with one command-line invocation per line, and the disBatch engine sends jobs to each node in a batch allocation until the nodes are filled. New tasks are dispatched as jobs finish and resources become available. The return code of each task is collected by the engine, and the results are recorded in a status file that can be used to retry failed jobs.

The post-processing performance depends on the disk speed of the OLCF Alpine file system, but a typical post-processing job takes from a few to a dozen node-hours, depending on whether or not the simulation box outputs full time slices.

[10] https://github.com/flatironinstitute/disBatch

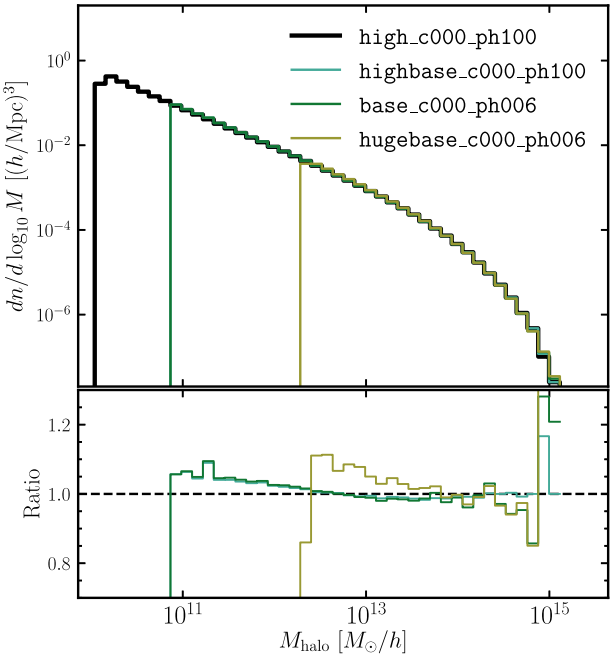

Figure 6. The halo mass function at $z = 0.5$ using the cleaned catalogues from four different simulations and three different mass resolutions, all for the `c000` cosmology. We present two phase-matched pairings: `highbase` compared with `high`, and then `hugebase` compared with `base`. The bottom panel shows the ratio of these curves with respect to the `high` resolution box. One sees that the two simulations with the same mass resolution, `base` and `highbase`, are in excellent agreement. When comparing a coarser simulation to a finer one, there is a mild excess of haloes at sizes below a few hundred particles.

5 COSMOLOGICAL OPPORTUNITIES WITH AbacusSummit

5.1 Scope

We have designed AbacusSummit to be superb at identifying haloes in the context of high-precision large-scale structure, with the intention of enabling wide-ranging studies of the clustering of galaxies and matter as a function of cosmological parameters.

The data volume and diversity is sufficiently large that it is difficult to summarize. One metric is the number of L1 haloes catalogued. Summing over all simulations to our catalogue limit of 35 particles, this totals 56.0 billion at $z = 2$, 68.4 billion at $z = 1.1$, and 67.3 billion at $z = 0.2$ (a slight decrease due to the smaller number of covariance boxes that reached $z = 0.2$). Restricting to haloes with more than 250 particles, about $5 \times 10^{10} h^{-1}\,\mathrm{M}_\odot$ in most simulations, we catalogue 5.7 billion at $z = 2$, 9.1 billion at $z = 1.1$, and 10.7 billion at $z = 0.2$.

Fig. 6 shows the mass function of haloes in each of our 3 mass resolutions, using the cleaned halo catalogues and the `c000` cosmology. One sees the obvious sign of the mass cut-off, but also a mild excess in the number of haloes at a given mass for halo sizes up to a few hundred particles. We note that this may be interpreted as the mass of a halo at coarse resolution being mildly higher than the same halo at the finer resolution, by roughly 5 per cent. Such dependence on the number of particles is not uncommon in halo finders and could result from a number of causes. Finer resolution gives more ability to detect and deblend neighbours or a better centring of the primary halo. It may also be that coarser resolution creates noise

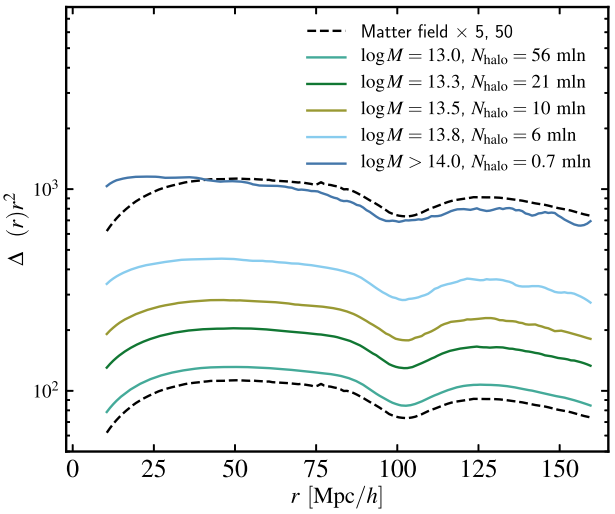

Figure 7. Modified correlation function of the haloes, $\Delta\xi(r) \equiv \bar{\xi}(<r) - \xi(r)$, for the $7.5h^{-1}$ Gpc huge box simulation at $z = 1.1$ as a function of scale, r. We split the haloes into 5 mass bins in $\log M$: [12.9, 13.1], [13.2, 13.4], [13.4, 13.6], [13.6, 14] and > 14. The average halo mass in the heaviest bin is $\log M \approx 14.15$ in units of $M_\odot/h$. The black dashed curves show the modified correlation function $\Delta\xi(r)$ computed for the matter field of the particles in subsample A and B and multiplied by a factor of 5 and 50 to aid the visual comparison.

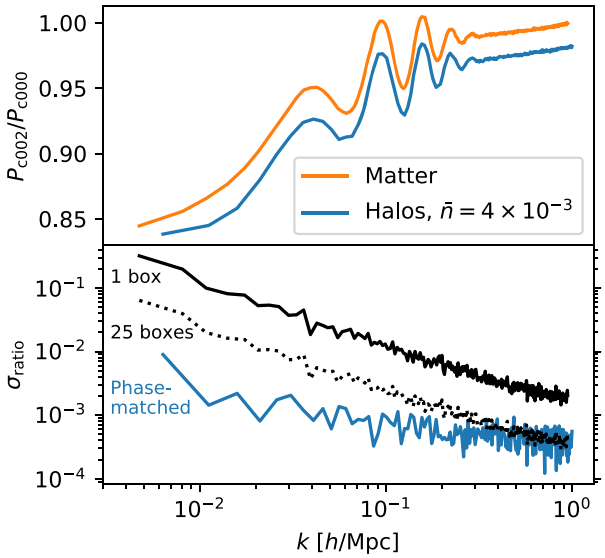

Figure 8. *Top panel:* The ratio of $z = 0.1$ power spectra from two phase-matched simulations that differ in cosmology – a derivative-like quantity. The halo samples are abundance matched to 4×10^{-3} h^3 Mpc^{-3}. *Bottom panel:* Sample variances on the halo power spectrum from 1 box (black line), 25 boxes (black dotted line), and the variance on the ratio of 1 phase-matched pair (blue line).

in the mass estimate, which given the steeply falling mass function tends to produce a small bias in the inferred mass. We conclude that applications depending on halo mass need to account for variations in definitions between different halo finding methods and resolutions.

Fig. 7 shows the large-scale correlation function of haloes in one of the $7.5h^{-1}$ Gpc huge box simulations. We use a variety of mass thresholds, starting at 200 particles, and compute the real-space two-point correlation function $\xi(r)$ of halo centres at $z = 1.1$. For pedagogical purposes, we show a modification

$$\Delta\xi(r) = \bar{\xi}(<r) - \xi(r) \tag{1}$$

where $\bar{\xi}(<r)$ is the average of ξ inside a sphere of radius r: $(3/r^3)\int_0^r \xi(r')r'^2dr'$. We note that $\Delta\xi(r)$ is related to the power spectrum $P(k)$ by

$$\Delta\xi(r) = \int_0^\infty j_2(kr)\frac{k^3P(k)}{2\pi^2}\frac{\mathrm{d}k}{k}. \tag{2}$$

A similar two-dimensional statistic $\Delta\Sigma$ has been common in the weak lensing literature (Bartelmann & Schneider 2001). As in that case, $\Delta\xi$ is unchanged by a constant offset in ξ and therefore has zero support at $k = 0$ and relative insensitivity to systematic errors at scales much larger than r. As with the usual pair-counting methods, it is unaffected by shot noise. However, it retains the localization of the acoustic scale information, here appearing as a dip rather than a peak. It also avoids a zero crossing and is linear in ξ, so a logarithmic scaling shows the multiplicative scaling of large-scale bias. A more complicated version of a statistic with this property was presented in Xu et al. (2010).

Fig. 7 shows the well-known progression of increasing bias with increasing halo mass Cole & Kaiser (1989), Mo & White (1996) as well as the near scale-independence of bias. At $z = 1.1$, the mass bin above $10^{14}h^{-1}$ $M_\odot$ is quite extreme, with a number density of only $1.6 \times 10^{-6}h^3$ Mpc^{-3} and a bias exceeding 7! These most massive haloes show some large-scale scale-dependence in their clustering bias as well as an increased width of the acoustic peak. These properties suggest care in interpreting the correlation function of high-mass clusters of galaxies.

One of the design goals of AbacusSummit is to provide the means to interpolate in a generous space of CDM cosmologies. The use of phase-matched simulations is a key part of that, as we expect that this will suppress the sample variance in many applications to be comparable to or smaller than the variance that comes from the 25 c000 base simulations. If so, one can interpolate to new cosmologies while retaining the sample variance of a $200h^{-3}$ Gpc^3 volume.

In Fig. 8, we show an example comparing c000 to c002. Both the matter power spectrum and halo power spectrum are shown, but we focus on the latter in the bottom panel as this is more representative of typical analysis. The fractional sample variance on the power spectrum is shown for $8h^{-3}$ Gpc^3 (one box; black line) and $200h^{-3}$ Gpc^3 (25 boxes; black dotted line), and the sample variance on the power spectrum ratio is shown for one $8h^{-3}$ Gpc^3 phase-matched pair (blue line). The phase-matched variance is smaller than the combined 25 boxes until about $k = 1.0h$Mpc^{-1}. At $k = 0.1h$Mpc^{-1}, the phase-matching is equivalent to more than 50 boxes. All measurements use $\Delta k = 0.005$ power spectrum binning.

5.2 Accuracy

A major opportunity with the Abacus code is its accuracy. The near field is solved with explicit $O(N^2)$ summation, with no approximation from trees, out to a distance of 2 Mpc. Meanwhile, the multipole method generates very accurate far field forces, with modest errors from cells two units away and rapidly reaching machine precision beyond. By comparing to control simulations in double-precision and with higher multipole order, we measure typical relative force errors of 10^{-5}, far more accurate than codes that use particle-mesh far fields.

AbacusSummit uses initial conditions with second-order Lagrangian perturbation theory, which is known to provide substantial improvements over the first-order theory (Zel'dovich approximation)

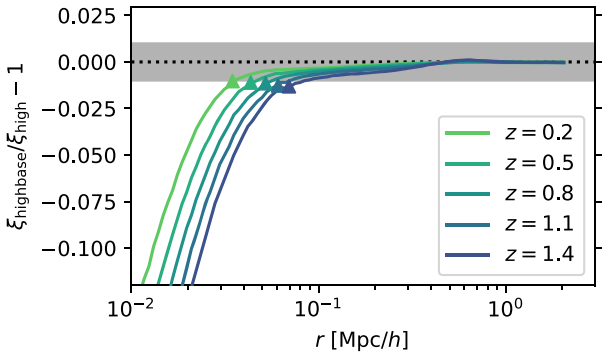

Figure 9. The small-scale matter correlation function of the `highbase` simulation (base mass resolution) compared with the `high` simulation (6 × better mass resolution). The triangles indicate the 4 × the comoving softening length of the `highbase` sim at that epoch (recalling that our softening is fixed in proper coordinates).

for rare density peaks (Crocce, Pueblas & Scoccimarro 2006; Reed et al. 2013; L'Huillier, Park & Kim 2014). Moreover, we use the method of Garrison et al. (2016) to correct for the systematic underevolution of continuum linear perturbation theory in a displaced discrete lattice. The correction persists through intermediate redshifts, yielding a few-per cent boost in the power near the Nyquist wavenumber of the particle lattice, matching what higher-resolution simulations yield.

The high speed of Abacus coupled with the modest force-softening of our application allows us to proceed simply with global time stepping, avoiding the potential errors of individualized time-steps. Garrison et al. (2018b) estimates that we have reached 1 per cent time-step convergence of the small-scale correlation function for our final answer.

To further leverage this opportunity, we use a spline softening law that returns to the exact $1/r^2$ force at finite radius, unlike the commonly used Plummer softening based on the potential $1/\sqrt{r^2+\epsilon^2}$, which converges only quadratically to the correct large-scale force law. This spline softening is moderately more expensive to compute, but the large hardware speed of the GPUs makes this an advantageous choice. Garrison et al. (2018b) shows the difference between these two softening choices, with Plummer softening showing per cent-level modifications of the $z = 0$ clustering even at 15ϵ.

As with any N-body simulation, there is a limit on spatial resolution due to the mass resolution of the particles and the force softening. We have explored this using scale-free simulations (Joyce et al. 2021; Leroy et al. 2021; Garrison et al. 2021c), where one evolves a simulation from a power-law initial power spectrum. These simulations show that a small force softening is not sufficient to reach a converged result for the small-scale correlation function; one is also limited by mass resolution, with a dependence that scales as $a^{-1/2}$. Our softening length is chosen to capture the available convergence given the particle resolution, as shown in Garrison et al. (2021c).

To further demonstrate this, we here compare the small-scale matter correlation function between our highest mass resolution and base mass resolution, using the phase-matched pair `highbase_c000_ph100` to `high_c000_ph100`, with particle mass $2 \times 10^9 h^{-1}\,{\rm M}_\odot$ and $3.5 \times 10^8 h^{-1}\,{\rm M}_\odot$, respectively. Fig. 9 shows the ratio of the correlation functions at five redshifts. We see that the two match to within 1 per cent to approximately 35 to 90 h^{-1} kpc between redshifts 0.2 and 1.4, decreasing with later epoch. The point at which the difference reaches 1 per cent is approximately given by $4\epsilon(a)$, where $\epsilon(a)$ is the comoving softening length, recalling our softening is fixed in proper coordinates to 7.2 h^{-1} kpc. This scaling appears to differ from the $a^{-1/2}$ scaling in Garrison et al. (2021c), although the mild change in scale factor in this plot makes the exact power-law ambiguous. Two contributing reasons may be (a) we have chosen a softening length that is comparable to the limit expected from the mass resolution, and (b) we do not integrate with an arbitrarily short time-step. For the latter, it is instructive to compute that time-steps of $10h^{-1}$ Myr imply that for a circular orbit in a high-mass cluster with circular velocity 1000 km s^{-1} would have $6\pi \approx 20$ leapfrog steps per orbit at a radius of $30h^{-1}$ proper kpc (about four times the softening length). Leapfrog integrations of two-body orbits with this number of steps do show reasonable conservation of radius, but develop a lag of about 10 deg in phase per orbit. The number of steps scales as radius divided by circular velocity, and integration accuracy scales as the inverse square of the number of steps. Given the sensitivity of the small-scale correlation function to high-mass haloes, it is not surprising that we start to see some mild variations at a spatial scale where the most massive systems require orbital times shorter than a few dozen time-steps.

The $z = 1.4$ correlation function also exhibits an excess of about 0.1 per cent at $0.6h^{-1}$ Mpc, an effect attributed to the 'memory' of the initial particle lattice. This is illustrated in Fig. 10, which shows a single square of 150^2 particles selected from a single Lagrangian sheet (a single particle plane in the initial conditions) at redshifts $z = 2.5$ and $z = 0.1$. The lattice is seen quite strongly at the earlier time, as most particles have not yet fallen into haloes or filaments and the deformations of the Lagrangian sheet are weak. At low redshift, most particles have fallen into dense structures, but the voids become even sparser, and the remnants of the initial condition lattice are visible.

We stress that this is endemic to any cosmological N-body simulation based on discrete particles; users must always take note that the regions that have not yet suffered any collapse will remember their initial configuration. Randomized initial configurations like a glass make the effect less visibly obvious, but it would still be present. Indeed, there are opportunities in uncollapsed regions to use the coldness of the initial sheet in phase space to estimate density and perform interpolations (e.g. Abel, Hahn & Kaehler 2012; Shandarin, Habib & Heitmann 2012); we note that the particle ID numbers in the full time-slice outputs could be helpful in such work.

As a final caveat, we note that while the N-body problem is crisply posed and we believe that Abacus offers very accurate solutions thereof, the identification of haloes and the associated interpretations of galaxy locations are always subject to choices. There will surely be differences between CompaSO haloes and those of other methods, and it is possible that these differences can affect mock galaxy catalogues in ways that matter for cosmological analysis. Users of any N-body simulation must consider this.

6 CONCLUSION

The AbacusSummit simulation suite constitutes a uniquely large and diverse data-set designed to enable a broad range of science applications in preparation for the next generation of large-scale structure cosmology. Simulating roughly 60 trillion particles, the suite constitutes the largest N-body data set produced to date. AbacusSummit has been designed to provide a large-volume, high-accuracy simulation of the Planck 2018 ΛCDM cosmology, as well as a grid of 96 other cosmologies to allow for interpolation in parameter space and emulator construction. AbacusSummit likewise includes 1883 small boxes to support covariance matrix estimation under periodic boundary conditions, and an array of simulations

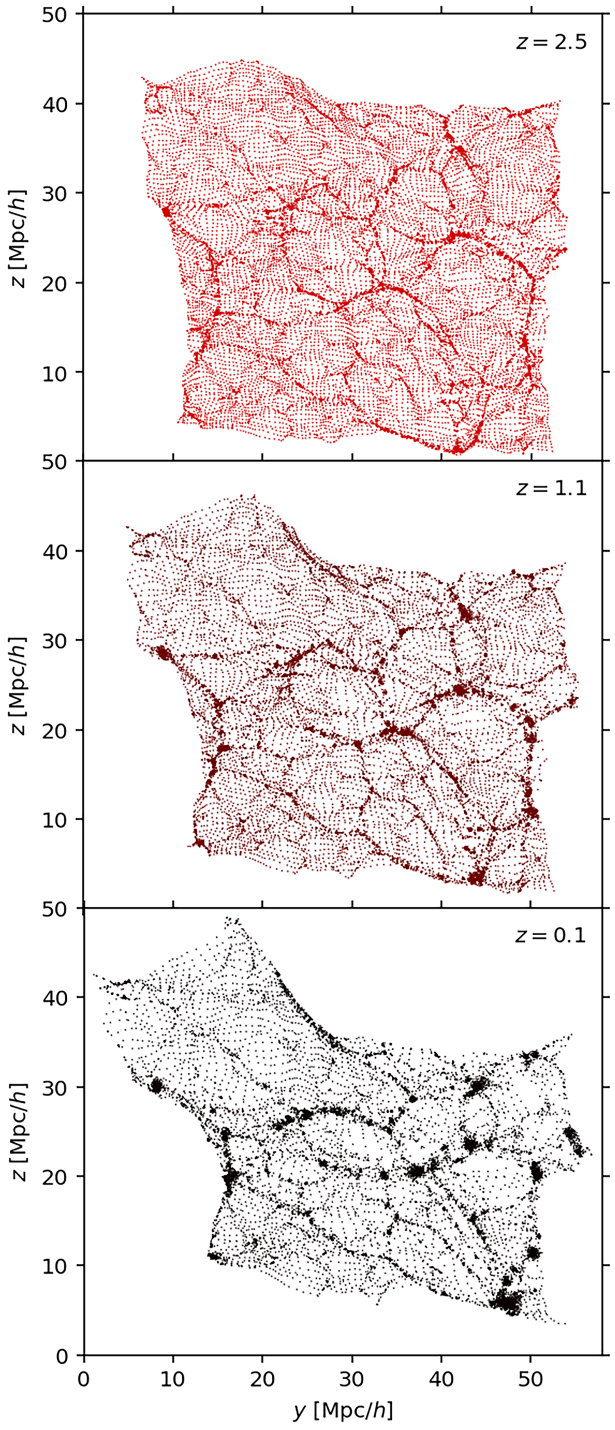

Figure 10. A single cutout of a Lagrangian plane – a square from a particle plane selected in the initial conditions – at redshifts $z = 2.5$, $z = 1.1$, and $z = 0.1$ from the `highbase` simulation. The 'memory' of the initial lattice configuration persists to low redshift in low-density regions.

designed to facilitate code comparison studies, group finding and mass resolution studies, as well as comparisons to major flagship simulations from other codes. AbacusSummit simulations include a diverse set of data-products which have undergone extensive validation, including particle subsamples, halo catalogues, merger trees, kernel density estimates, light cones, and projected density maps of the light cones. AbacusSummit was produced using the Abacus code and is available publicly at a DOI link published on https://abacussummit.readthedocs.io/.

Software: astropy (Astropy Collaboration et al. 2018), numpy (van der Walt, Colbert & Varoquaux 2011), (Virtanen et al. 2020), numba (Lam, Pitrou & Seibert 2015), cuda (Nickolls et al. 2008), Intel TBB (Reinders 2007), matplotlib (Hunter 2007), ASDF (Greenfield et al. 2015), Globus (Foster 2011; Allen et al. 2012), corner.py (Foreman-Mackey 2016), corrfunc (Sinha & Garrison 2019, 2020).

ACKNOWLEDGEMENTS

We thank Philip Pinto for his advice on the project and Peter Behroozi, Shaun Cole, Salman Habib, Katrin Heitmann, Alina Kiessling, and Joachim Stadel for helpful conversations.

This work has been supported by NSF AST-1313285, DOE-SC0013718, and NASA ROSES grant 12-EUCLID12-0004. DJE is supported in part as a Simons Foundation investigator. NAM was supported in part as an NSF Graduate Research Fellow. LHG is supported by the Centre for Computational Astrophysics at the Flatiron Institute, which is supported by the Simons Foundation. SB is supported by Harvard University through the ITC Fellowship.

This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725. Computation of the merger trees used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility located at Lawrence Berkeley National Laboratory, operated under Contract No. DE-AC02-05CH11231. The AbacusSummit simulations have been supported by OLCF projects AST135 and AST145, the latter through the Department of Energy ALCC programme.

We would like to thank the OLCF and NERSC staff for their highly responsive and expert assistance, both scientific and administrative, during the course of this project.

DATA AVAILABILITY

With this paper, we are placing the AbacusSummit simulation suite into the public domain, subject to the academic citations described at https://abacussummit.readthedocs.io/en/latest/citation.html.

Data access is available through OLCF's Constellation portal. The persistent DOI describing the data release is 10.13139/OLCF/1811689. Instructions for accessing the data are given at https://abacussummit.readthedocs.io/en/latest/data-access.html.

REFERENCES

Abel T., Hahn O., Kaehler R., 2012, MNRAS, 427, 61
Aghanim N. et al., 2018, Available at: https://arxiv.org/abs/1807.06209
Aihara H. et al., 2018, PASJ, 70, S4
Allen B. et al., 2012, Commun. ACM, 55, 81
Angulo R. E., Pontzen A., 2016, MNRAS, 462, L1
Angulo R. E., Springel V., White S. D. M., Jenkins A., Baugh C. M., Frenk C. S., 2012, MNRAS, 426, 2046
Astropy Collaboration et al., 2018, AJ, 156, 123
Bartelmann M., Schneider P., 2001, Phys. Rep., 340, 291
Behroozi P. S., Wechsler R. H., Wu H.-Y., 2012, ApJ, 762, 109
Bose S., Eisenstein D., Hadzhiyska B., Garrison L., Yuan S., Maksimova N., 2021, submitted
Bryan G. L., Norman M. L., 1998, ApJ, 495, 80
Calabrese E. et al., 2017, Phys. Rev. D, 95, 063525

Cole S., Kaiser N., 1989, MNRAS, 237, 1127
Crocce M., Pueblas S., Scoccimarro R., 2006, MNRAS, 373, 369
Crocce M., Fosalba P., Castander F. J., Gaztañaga E., 2010, MNRAS, 403, 1353
Davis M., Efstathiou G., Frenk C. S., White S. D. M., 1985, ApJ, 292, 371
DeRose J. et al., 2019, ApJ, 875, 69
Dubois Y., Peirani S., Pichon C., Devriendt J., Gavazzi R., Welker C., Volonteri M., 2016, MNRAS, 463, 3948
Emberson J. D. et al., 2017, Res. Astron. Astrophys., 17, 085
Farr W. M., Bertschinger E., 2007, ApJ, 663, 1420
Foreman-Mackey D., 2016, J. Open Source Softw., 1, 24
Foster I., 2011, IEEE Internet Comput., 15, 70
Garrison L. H., Eisenstein D. J., Ferrer D., Metchnik M. V., Pinto P. A., 2016, MNRAS, 461, 4125
Garrison L. H., Eisenstein D. J., Ferrer D., Tinker J. L., Pinto P. A., Weinberg D. H., 2018a, ApJS, 236, 43
Garrison L. H., Eisenstein D. J., Pinto P. A., 2019, MNRAS, 485, 3370
Garrison L., Eisenstein D., Ferrer D., Maksimova N., Pinto P., 2021a, MNRAS, Available at:https://ui.adsabs.harvard.edu/abs/2021MNRAS.tmp.2276G/abstract
Garrison L. H., Eisenstein D. J., Maksimova N. A., 2021b, First International Symposium on Checkpointing for Supercomputing, Available at:https://arxiv.org/abs/2102.13140
Garrison L. H., Joyce M., Eisenstein D. J., 2021c, MNRAS, 504, 3550
Greenfield P., Droettboom M., Bray E., 2015, Astron. Comput., 12, 240
Greengard L., Rokhlin V., 1987, J. Comput. Phys., 73, 325
Habib S. et al., 2016, New Astron., 42, 49
Hadzhiyska B., Eisenstein D., Bose S., Garrison L., Maksimova N., 2021, submitted
Heitmann K., Higdon D., White M., Habib S., Williams B. J., Lawrence E., Wagner C., 2009, ApJ, 705, 156
Heitmann K. et al., 2019, ApJS, 244, 17
Hunter J. D., 2007, Comput. Sci. Eng., 9, 90
Ishiyama T. et al., 2020, preprint (arXiv:2007.14720)
Joyce M., Garrison L., Eisenstein D., 2021, MNRAS, 501, 5051
Kim J., Park C., Gott, J. Richard I., Dubinski J., 2009, ApJ, 701, 1547
Klypin A., Yepes G., Gottlöber S., Prada F., Heß S., 2016, MNRAS, 457, 4340
Klypin A., Yepes G., Gottlöber S., Prada F., Heß S., 2016, MNRAS, 457, 4340
L'Huillier B., Park C., Kim J., 2014, New Astron., 30, 79
Lacey C., Cole S., 1994, MNRAS, 271, 676
Lam S. K., Pitrou A., Seibert S., 2015, in Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC, LLVM '15. Association for Computing Machinery, New York
Laureijs R. et al., 2011, Available at:https://arxiv.org/abs/1110.3193
Leroy M., Garrison L., Eisenstein D., Joyce M., Maleubre S., 2021, MNRAS, 501, 5064
Lesgourgues J., 2011, The Cosmic Linear Anisotropy Solving System (CLASS) I: Overview. Available at: https://arxiv.org/abs/1104.2932
Levi M. et al., 2013, Available at: https://arxiv.org/abs/1308.0847
Liu J., Bird S., Zorrilla Matilla J. M., Hill J. C., Haiman Z., Madhavacheril M. S., Petri A., Spergel D. N., 2018, J. Cosmol. Astropart. Phys., 2018, 049
Lorenz C. D., Ziff R. M., 2001, J. Chem. Phys., 114, 3659
LSST Science Collaboration et al., 2009, preprint (arXiv:e-print)
Metchnik M. V., 2009, PhD thesis, Univ. Arizona
Mo H. J., White S. D. M., 1996, MNRAS, 282, 347
Nelson D. et al., 2019, The IllustrisTNG Simulations: Public Data Release. Available at: https://arxiv.org/abs/1812.05609
Nickolls J., Buck I., Garland M., Skadron K., 2008, Queue, 6, 40
Nishimichi T. et al., 2019, ApJ, 884, 29
ORNL, 2020, Summit User Guide. Available at: https://docs.olcf.ornl.gov/systems/summit
Potter D., Stadel J., 2016, PKDGRAV3: Parallel Gravity Code. Available at: http://ascl.net/1609.016
Potter D., Stadel J., Teyssier R., 2017, Comput. Astrophys. Cosmol., 4, 2
Quinn T., Katz N., Stadel J., Lake G., 1997, Available at:https://arxiv.org/abs/astro-ph/9710043
Reed D. S., Smith R. E., Potter D., Schneider A., Stadel J., Moore B., 2013, MNRAS, 431, 1866
Reinders J., 2007, Intel Threading Building Blocks, 1st edn. O'Reilly & Associates, Inc., USA
Shandarin S., Habib S., Heitmann K., 2012, Phys. Rev. D, 85, 083005
Sinha M., Garrison L. H., 2019, Software Challenges to Exascale Computing. Second Workshopp. 3
Sinha M., Garrison L. H., 2020, MNRAS, 491, 3022
Skillman S. W., Warren M. S., Turk M. J., Wechsler R. H., Holz D. E., Sutter P. M., 2014, Dark Sky Simulations: Early Data Release. Available at: https://arxiv.org/abs/1407.2600
Spergel D. et al., 2015, Available at:https://arxiv.org/abs/1503.03757
Tamura N. et al., 2016, Proc. SPIE, 9908, 99081M
Tinker J., Kravtsov A. V., Klypin A., Abazajian K., Warren M., Yepes G., Gottlöber S., Holz D. E., 2008, ApJ, 688, 709
van der Walt S., Colbert S. C., Varoquaux G., 2011, Comput. Sci. Eng., 13, 22
Villaescusa-Navarro F. et al., 2020, ApJS, 250, 2
Virtanen P. et al., 2020, Nature Methods, 17, 261
Weinberg M. D., Katz N., 2007, MNRAS, 375, 425
Xu X. et al., 2010, ApJ, 718, 1224

This paper has been typeset from a TEX/LATEX file prepared by the author.